OSNOVE OPERACIJSKEGA SISTEMA DELTA/M

Učbenik za seminar M150

Verzija 1.2

računalniški sistemi delta

NŠA

OSNOVE OPERACIJSKEGA SISTEMA

DELTA/M

Učbenik za seminar M150

Verzija 1.2

Ljubljana, september 1984

IskraDelta

PREDGOVOR

Priročnik je pisan kot spremljajoči tekst k seminarju OSNOVE OPERACIJSKEGA SISTEMA DELTA/M. V njem so razloženi osnovni pojmi in karakteristike operacijskega sistema DELTA/M ter napotki za njegovo uporabo. Priročnik je pisan po fonkcionalnih temah. Prvo poglavje obravnava ojnovne pojme s področja stojne in programske opreme. Drugo poglavje je namenjenu delo s terminalom in ukazom, ki se nanašajo na terminal. Tretje poglavje e najobsežnejše in pojasnuje FILES-11 strukturo ter delo z datotekami: kreiranjle, prepisovanje, brisanje, listanje direktorijev. Delo z enotami je opisano v četrtem poglavju, razvoj programov in njihovo izvajanje pa v petem. Posredne ukazovne datoteke so tema šestega poglavja. Sedmo in osmo poglavje na kretko obravnavata PRINT in FLX pomožna programa.

Ker je priročnik namenjen seminarju, ni popoln zbir ukazov. Kar manjka, lahko najdete v MCR Operations Manualu ali pa v Utilities References Manualu.

Slavko Lenarčič

računalniški sistemi delta

M150 - OSNOVE OPERACIJSKEGA SISTEMA DELTA/M

KAZALO

Vsebina Stran

POGLAVJE3 DELO Z DATOTEKAMI

r

POGLAVJE6 POSREDNE UKAZOVNE DATOTEKE

POGLAVJE1

UVOD

Računalnik je sestavljen iz dveh komponent:strojne opreme (hardware) in programske opreme (software).

Strojna oprema so vsi vidni sestavni deli: printerji, diskovne enote, tračne enote, disketne enote, terminali, procesorji in kup žic, ki vse to povezujejo.

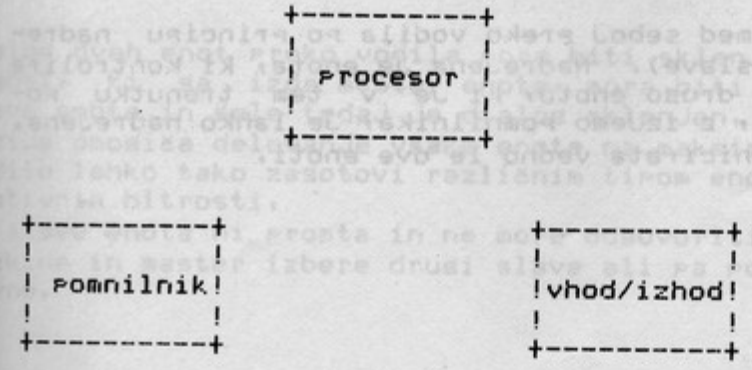
Programska oprema je skupina programov, ki omogočajo ljudem delo z računalnikom.

V uvodu si bomo ogledali osnovne značilnosti strojne opreme računalnikov DELTA z 16-bitnimi procesorji ter pojasnili nekatere osnovne pojme.

1.1 ARHITEKTURA STROJNE OPREME SISTEMOV DELTA

Strojno opremo računalnikov lahko razdelimo na tri dele:

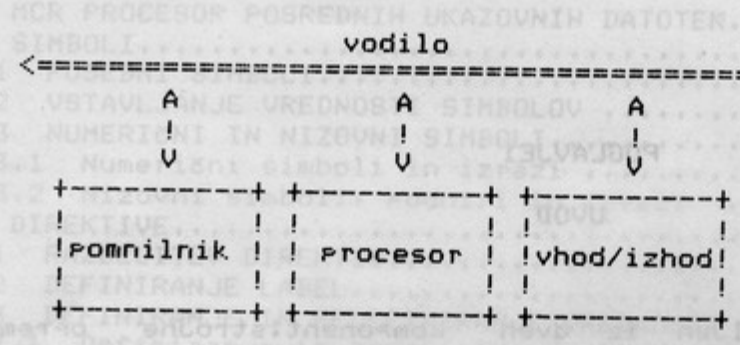
- procesor za aritmetične in logične operacije ter kontrolo
- pomnilnik za shranjevanje programov in podatkov
- vhodno/izhodne enote za komunikacijo med človekom in računalnikom



sl 1. osnovni elementi računalnika

Pri sistemih DELTA povezuje vse funkciunalne enote en kanal imenovan VODILO ali UNIBUS. Ker je kanal skupen, lahko vsak element

komunicira s katerim koli elementom, ki je prikjučen na ta kanal. Shematsko lahko to prikažemo tako:



sl 2. Povezovalni kanal pri DELTA sistemih

Kakor vidimo, vodilo povezuje vse enote:

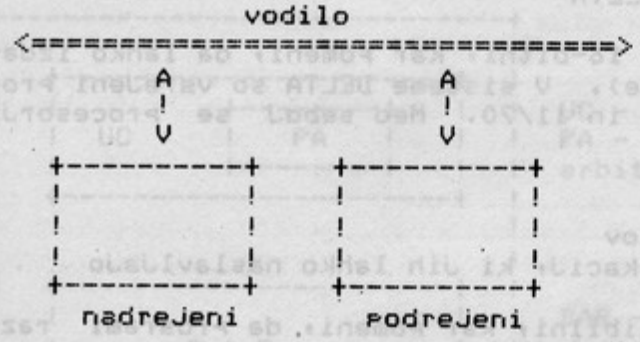
- pomnilnik z vhodno/izhodnimi enotami
- procesor z vhodno/izhodnimi enotami
- procesor s pomnilnikom

1.1.1 UNIBUS

Vodilo predstavja 56 signalnih linij, na katere so paralelno priključene vse enote. 51 linij je dvosmernih, 5 pa enosmernih. Maksimalna propustnost vodila je ena 16-bitna beseda vsakih 400 nano sekund ali približno 2,5 miljona 16-bitnih besed na sekundo. Ker so enote paralelno vezane na vodilo, lahko enote po potrebi kontinuirano dodajamo. Enote delujejo asinhrono (vsaka s svojo hitrostjo).

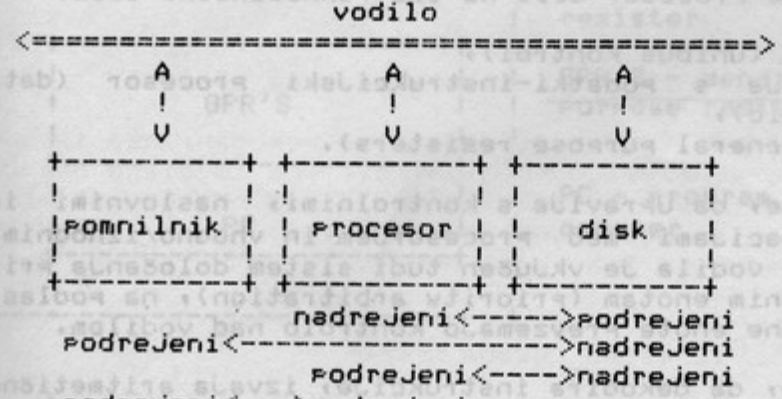
1.1.1.1 Relacija nadrejeni-podrejeni

Dve enoti komunicirata med seboj preko vodila po principu nadrejeni-podrejeni (master-slave). Nadrejena je enota, ki kontrolira vodilo, ko komunicira z drugo enoto, ki je v tem trenutku podrejena. Vsaka enota, z izjemo pomnilnika, je lahko nadrejena. Preko vodila lahko komunicirata vedno le dve enoti.



sl 3. relacija nadrejeni-podrejeni

Relacija nadrejeni-podrejeni je dinamična; to pomeni da enota, ki je bila nadrejena, prepusti kontrolo drugi, ki postane nadrejena. Tako je lahko neka enota v določenem trenutku nadrejena v naslednjem pa podrejena.



sl 4. primer dinamične kontrole nad vodilom

Dialog dveh enot preko vodila mora biti sklenjen. Vsak kontrolni signal, ki sa izda master enota, mora biti potrjen z odgovorom slave enote in šele tedaj je dialog sklenjen. Zaključena komunikacija omogiča delovanje vsake enote na maksimalni hitrosti. Vodilo lahko tako zagotovi različnim tipom enot širok spekter operativnih hitrosti.
Če slave enota ni prosta in ne more odgovoriti, se komunikacija prekine in master izbere drugi slave ali pa počaka in poskusi ponovno.

1.1.2 PROCESORJI SISTEMOV DELTA

Procesorji sistemov DELTA so 16-bitni, kar pomeni, da lahko izdajajo 16-bitne naslove (adrese). V sisteme DELTA so vgrajeni procesorji 11/24, 11/34A, 11/44 in 11/70. Med seboj se procesorji razlikujejo po:

- velikosti nabora ukazov
- hitrosti izvajanja ukazov
- številu pomnilniških lokacij, ki jih lahko naslavljajo

Procesorji so navzgor kompatibilni, kar pomeni, da programi razviti za manjše procesorje delujejo enako dobro tudi na večjih. Obratna relacija ni možna.

1.1.2.1 Organizacija procesorja

Pri sistemih DELTA se procesor deli na tri funkcionalne dele:

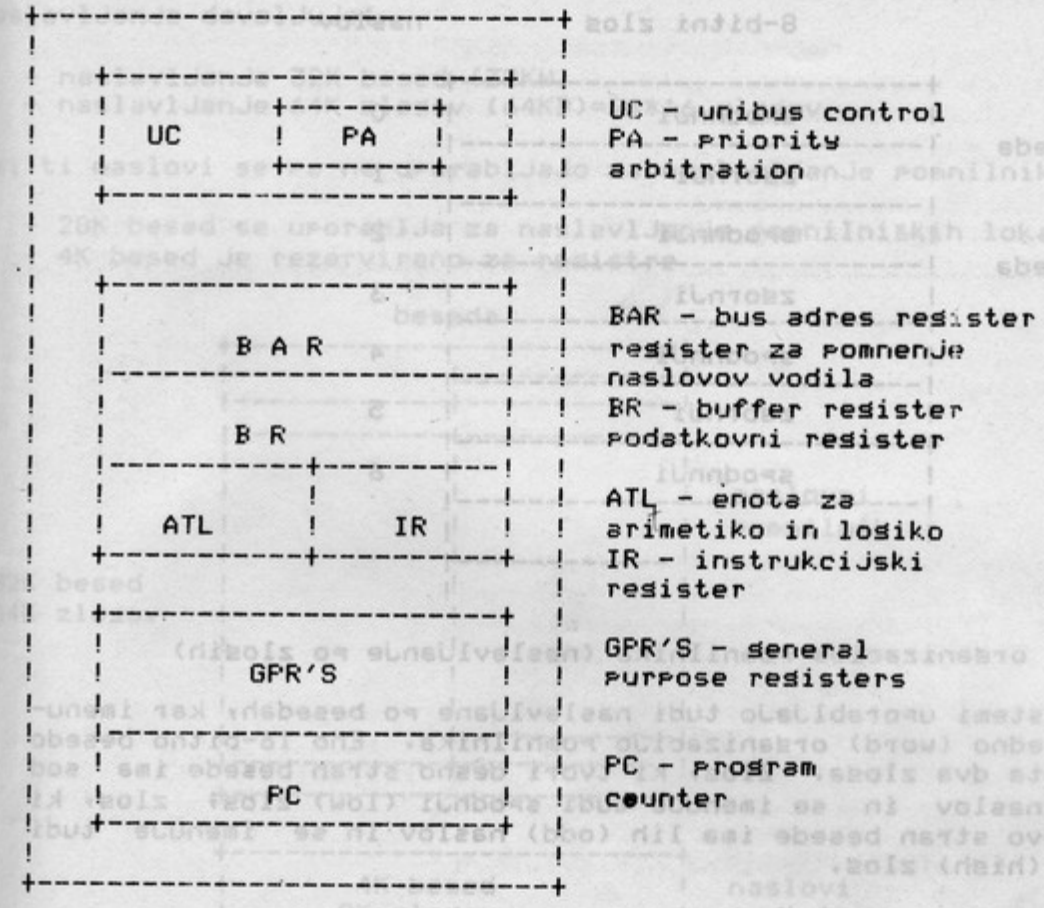
- kontrola vodila (unibus kontrol),
- logiko operiranja s podatki-instrukcijski procesor (data manipulatio logic),
- registri (GPR-general purpose registers).

Naloga prvega dela je, da upravlja s kontrolnimi, naslovnimi in podatkovnimi informacijami med procesorjem in vhodno/izhodnimi enotami. V kontrolo vodila je vkjučen tudi sistem določanja prioritet vhodno/izhodnim enotam (priority arbitration), na podlagi katerih vhodno/izhodne enote prevzemajo kontrolo nad vodilom.

Drugi del ima nalogo, da dekodira instrukcije, izvaja aritmetične operacije in kodira vse aktivnosti procesorja.

Tretji del - registri, služijo kot interni pomnilni elementi in so lahko operandi, naslovi operandov, rezultati (akomulaturji), kazalci (pointerji) ali števci (counters).

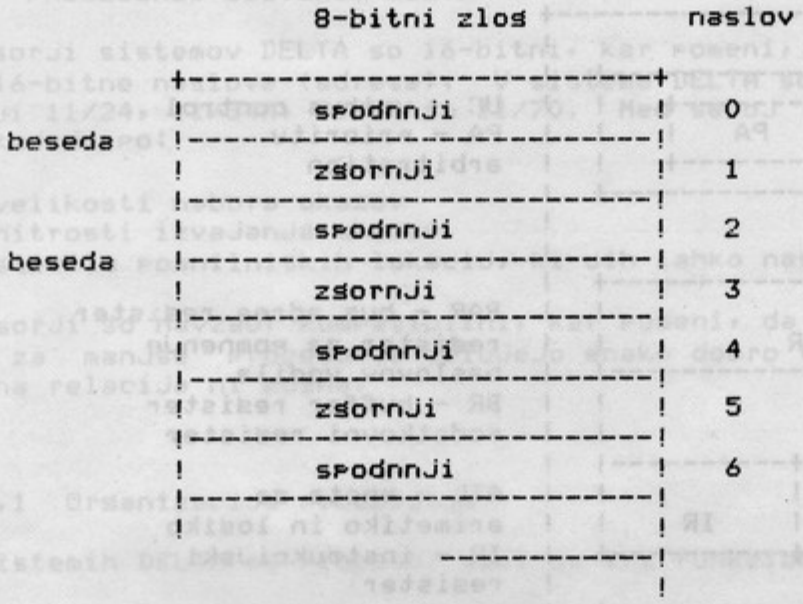
DELTA 340 ima 8, DELTA 700 pa 16 takih registrov. Registri so 16-bitni.



sl 6. organizacija procesorja

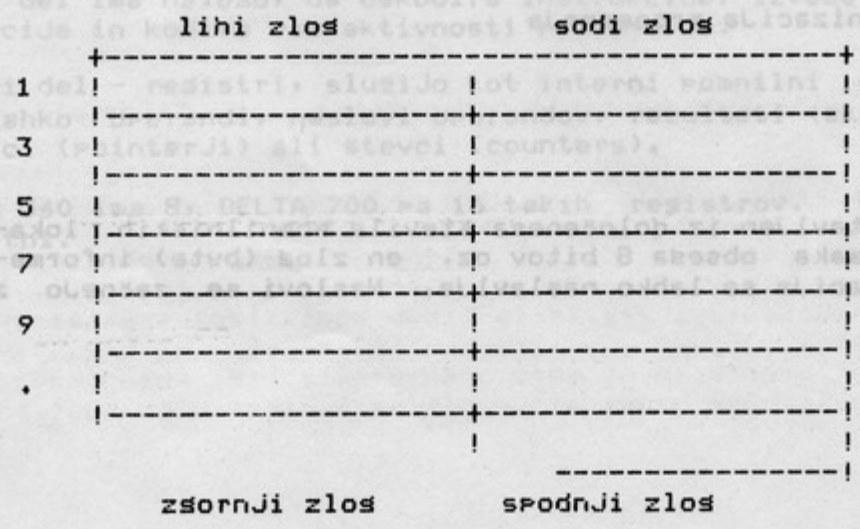
1.1.3 POMNILNIK

Pomnilnik je sestavljen iz določenega števila pomnilniških lokacij, katerih vsaka obsega 8 bitov oz. en zlog (byte) informacije. Vsaka lokacija se lahko naslavlja. Naslovi se začnejo z nič.



sl 7. organizacija pomnilnika (naslavljanje po zlogih)

DELTA sistemi uporabljajo tudi naslavljane po besedah, kar imenujemo besedno (word) organizacijo pomnilnika. Eno 16-bitno besedo oblikujeta dva zloga. Zlog, ki tvori desno stran besede ima sod (even) naslov in se imenuje tudi spodnji (low) zlog; zlog, ki tvori levo stran besede ima lih (odd) naslov in se imenuje tudi zgornji (high) zlog.



sl 8. besedna organizacija pomnilnika

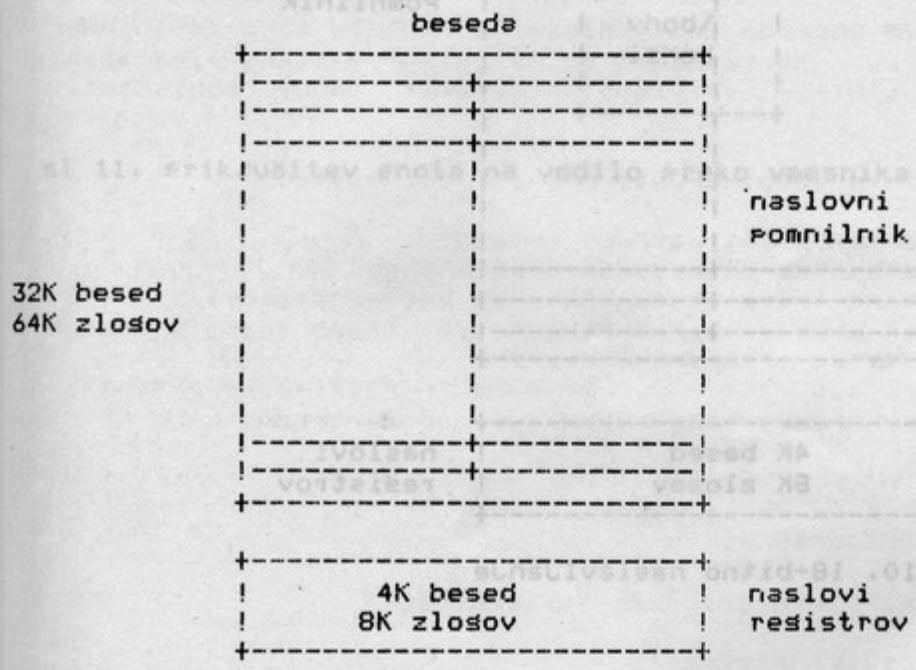
Taka naslovna struktura omogoča uporabniku dostop samo do spodnjega ali samo do zgornjega zloga, ali pa do obeh.

Sistem DELTA uporablja 16-bitno naslovno strukturo. 16-bitno naslavljanje dovoljuje:

- naslavljanje 32K besed (32KW)
- naslavljanje 64K zlogov (64KB)=2**16 zlogov

Vsi ti naslovi se pa ne uporabljajo za naslavljanje pomnilnika:

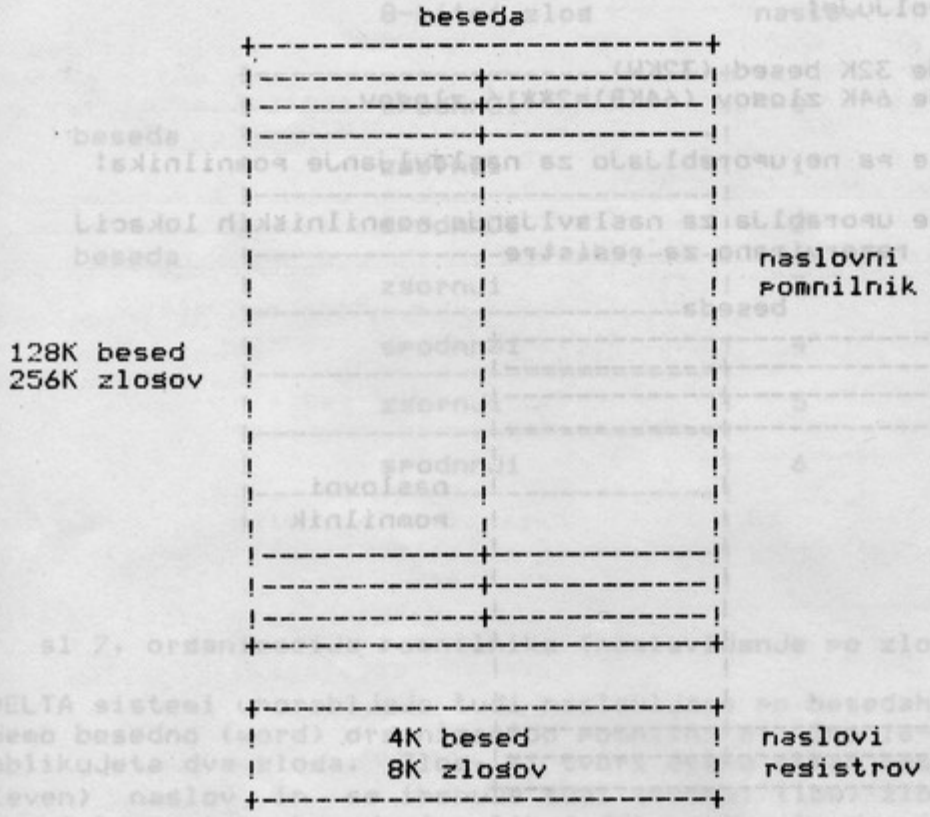
- 28K besed se uporablja za naslavljanje pomnilniških lokacij
- 4K besed je rezervirano za registre



sl 9. 16-bitno naslavljanje

Poseben hardwareski instrument, ki se imenuje memory management omogoča tudi 18-bitno oz. 22-bitno naslavljanje. Na ta način je omogočeno naslavljanje preko 64KB. Razširjeni 18-bitni naslov omogoča naslavljanje do 2**18 = 262.144 zlogov= 256 KB glavnega pomnilnika, z 22-bitnim naslovom pa je možno naslavljati maksimalno 2**22 = 4.194.304 zlogov ali preko 4 mega zlogov glavnega pomnilnika.

Sistemi DELTA 340 so oblikovani za 18-bitno naslavljanje, sistemi DELTA 400, DELTA 622, DELTA 700 in DELTA 800 pa za 22-bitno naslavljanje. Razširjeno naslavljanje je možno, če je sistem opremljen z memory management enoto.



sl 10. 18-bitno naslavljanje

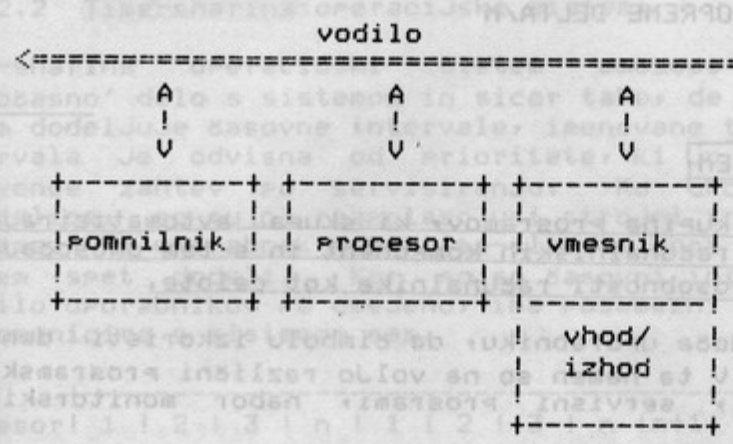
1.1.4 VHODNO/IZHODNE ENOTE

Na DELTA sisteme lahko prikjučimo različne vhodno/izhodne enote in sicer:

- počasne vhodno/izhodne enote: teleprinterji, čitalci in luknjači papirnatega traku, čitalci kartic, vrstični tiskalniki ter zaslonski terminali.

- hitre enote (mass storage divices): diskovne in magnetnotračne enote ter disketne enote.

Vhodno/izhodne enote ne morejo biti prikjučene direktno na vodilo, temveč so prikjučene preko vmesnika (interface unit), ki usklajuje hitrosti signalov na vodilu in posamezni enoti. Ker se enote med seboj razlikujejo, ne morejo uporabljati enakega vmesnika, ampak vsak tip enote uporablja drugačen vmesnik.
Vmesnik pretvarja informacije vodila v podatkovne in kontrolne signale za posamezno enoto.



sl 11. prikjučitev enote na vodilo preko vmesnika

1.2 OSNOVE PROGRAMSKE OPREME DELTA/M

1.2.1 OPERACIJSKI SISTEM

Operacijski sistem je skupina programov, ki skupaj avtomatizirajo in upravljajo delovanje računalniških komponent in s tem omogočajo čimboljše operativne sposobnosti računalnika kot celote.

Operacijski sistem omogoča uporabniku, da čimbolj izkoristi dane možnosti hardwarea. V ta namen so na voljo različni programski produkti: prevajalniki, servisni programi, nabor monitorskih ukazov ...

Zahteve, ki jih mora izpolnjevati operacijski sistem:

- čimbolj mora zaposliti strojno opremo
- omogočiti istočasno delo večim uporabnikom
- multiprograming (hkratno obdelovanje več programov)
- vzdrževati in upravljati podatkovno strukturo (delo z datotekami in vodenje spiskov datotek)
- izvajati kontrolo prioritet (strojne in programske opreme)
- registriranje opravljenega dela (job accounting).

1.2.2 TIPI OPERACIJSKIH SISTEMOV

Najbolj pogosti tipi operacijskih sistemov so:

- batch operacijski sistemi
- time-sharing operacijski sistemi
- real-time operacijski sistemi
- multi-programing operacijski sistemi

1.2.2.1 Batch operacijski sistemi

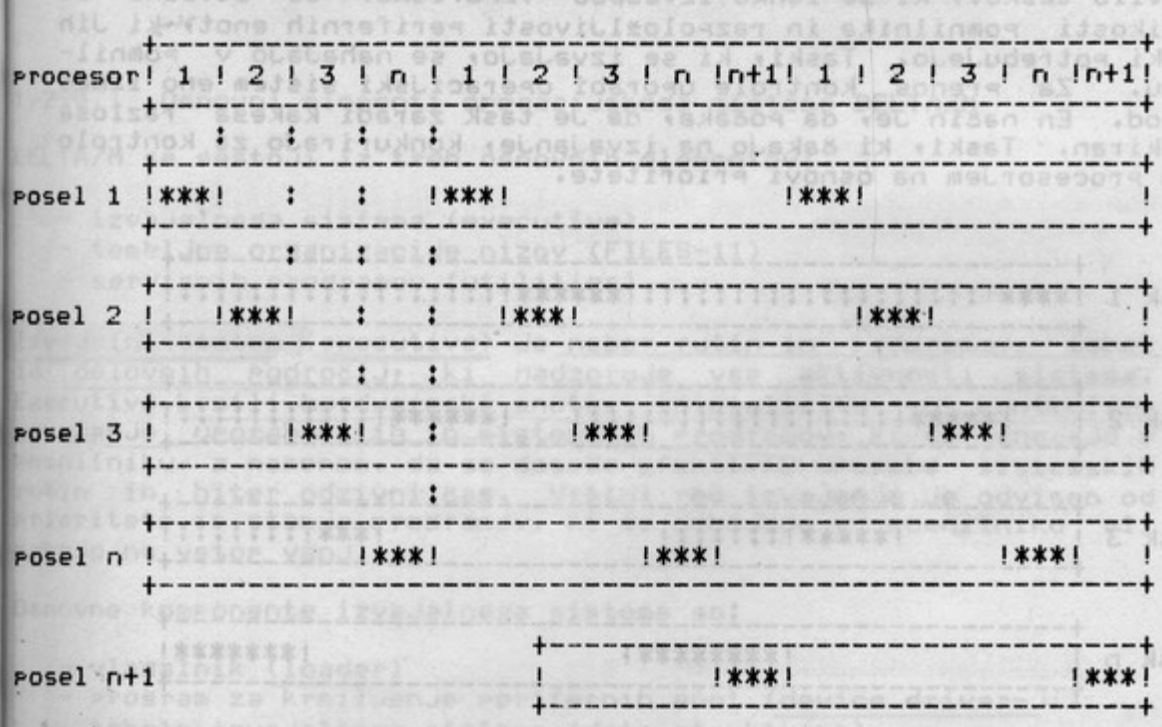
Za batch operacijske sisteme je značilno, da se izvajanje programov vrši zaporedno. Programi so naloženi v 'paket', iz katerega se polnijo v pomnilnik po sistemu 'eden ven, drugi noter'. Uporabnik lahko izvaja svoje progame-taske na dva načina:

- interaktivno, ko sam z ukazi predpiše potek izvajanja preko terminala
- s pomočjo datoteke, ki vsebuje vse ukaze s katerimi je določen potek izvajanja.

Pri drugem načinu je možnost napak manjša, saj odpadejo ročni posegi. Pri večini današnjih sistemov je batch način izvajanja programov združen z multiprogramiranjem.

1.2.2.2 Time-sharing operacijski sistemi

Time-sharing operacijski sistem omogoča več uporabnikom 'istočasno' delo s sistemom in sicer tako, de prikljućenim uporabnikom dodeljuje časovne intervale, imenovane time-slice. Dolžina intervala je odvisna od prioritete, ki jo ima uporabnik in od frekvence zahtev po servisiranju. Ko uporabnik dobi svoj time-slice, so mu na razpolago vsi strojni in programski resursi sistema. Ko uporabnik nima time-slicea, mora počakati, da mu sa sistem spet dodeli. Ker pa so časovni intervali zelo kratki, število uporabnikov pa omejeno, ima posamezni uporabnik občutek, de komunicira s sistemom sam.



sl 12. time-sharing

1.2.2.3 Real-time operacijski sistemi

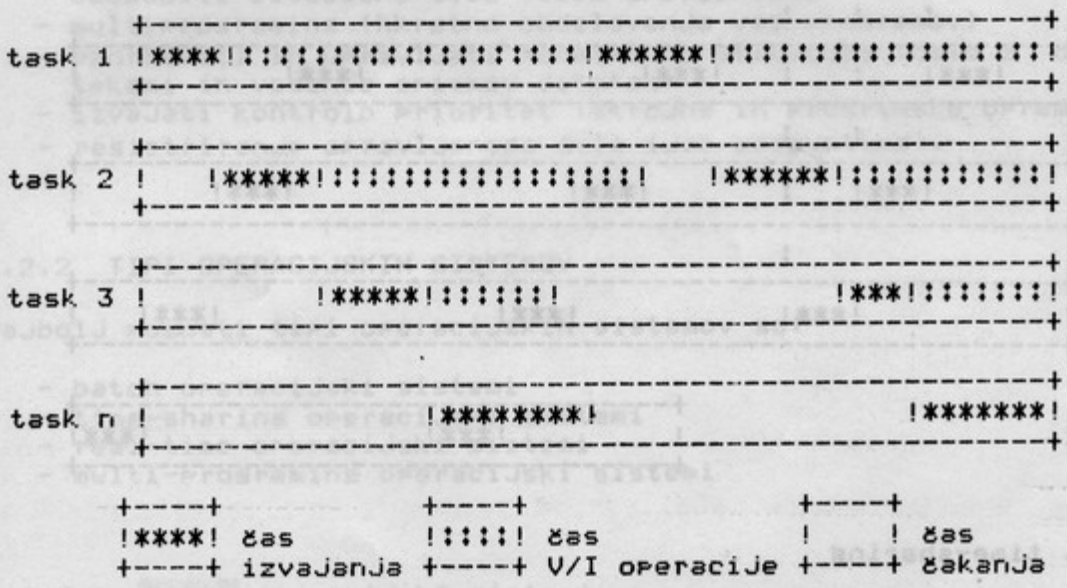
Real-time operacijski sistem omogoča delo v realnem času. Realni čas pa je lahko dokaj različen in je odvisen od procesa oziroma aplikacije. Za nekatere procese je to rang milisekunde, za druge sekunda, nekaj sekund, minuta ali celo ura.

Real-time operacijske sisteme srečamo najpogosteje pri regulacijah različnih procesov (kemičnih, nuklearnih, laboratorijskih ...), kjer nek pomemben dogodek v procesu zahteva takojšnje ukrepanje.

1.2.2.4 Multi-programing operacijski sistemi

V določenem trenutku procesor izvaja samo eno instrukcijo, kar pomeni, da se v določen trenutku izvaja le en program. Ker pa procesor izvaja aritmetične in logične operacije mnogo hitreje kot se izvajajo vhodno/izhodne operacije, prenese sistem kontrolo na drug program, ko ima prvi vhodno/izhodno operacijo. Tako je procesor bolje izkoriščen. Tak način dela imenujemo multi-programiranje in sistem multi-programing operacijski sistem.

Število taskov, ki se lahko izvajajo vzporedno, je odvisno od velikosti pomnilnika in razpoložljivosti perifernih enot, ki jih taski potrebujejo. Taski, ki se izvajajo, se nahajajo v pomnilniku. Za prenos kontrole uporabi operacijski sistem eno izmed metod. En način je, da počaka, da je task zaradi kakega razloga blokiran. Taski, ki čakajo na izvajanje, konkurirajo za kontrolo nad procesorjem na osnovi prioritete.



sl 13. multi-programing

1.2.3 DELTA/M OPERACIJSKI SISTEM

DELTA/M operacijski sistem omogoča delo v realnem času, batch, time-sharing in multi-programiranje in je zgrajen za serijo 16 bitnih računalnikov DELTA. DELTA/M je modificiran in dopolnjen operacijski sistem RSX-11M firme DIGITAL EQUIPMENT CORPORATION (DEC). Je diskovno orjentiran operacijski sistem. Usmerjen je predvsem v hitro odzivnost in v delo v realnem času. Multiprogramiranje na tem računalniku omogoča, da se delo v real-

nem času lahko kombinira z delom na razvijanju programov in z drugimi manj nujnimi uporabniškimi aplikacijami.Po eni strani se DELTA/M lahko uporablja kot samostojen krmilnik procesov, po drugi strani pa je to lahko večuporabniški sistem, namenjen predvsem razvijanju uporabniških aplikacij in njihovem izvajanju.

Pri operacijskem sistemu lahko izbiramo med mnogimi uslužnostnimi in pomožnimi programi.Nekateri med njimi so avtomatsko dostopni kot osnovni sestavni deli tega operacijskega sistema,drugi so dosegljivi samo s posebnimi postopki.Za vsako instalacijo se formira posebna verzija operacijskega sistema DELTA/M, glede na obstoječo konfiguracijo in glede na namen sistema.

1.2.3.1 Osnovni elementi operacijskega sistema DELTA/M

DELTA/M se sestoji iz treh osnovnih elementov:

- izvajalnega sistema (executive)
- temeljne organizacije nizov (FILES-11)
- servisnih programov (utilities)

Izvajalni sistem (executive) je nabor rutin in programov, tabel in delovnih področij, ki nadzoruje vse aktivnosti sistema. Executive krmili hardwareske enote, inicializira in kontrolira izvajanje uporabniških in sistemskih programov, ki se nahajajo v pomnilniku, z namenom, da se doseže efektivna uporaba sistemskih rutin in hiter odzivni čas. Vrstni red izvajanja je odvisen od prioritete in stanja programov, ki se nahajajo v pomnilniku ali čakajo na vstop vanj.

Osnovne komponente izvajalnega sistema so:

- vlagalnik (loader)
- program za krmiljenje perifernih enot (device driver-ji)
- tabele izvajalnega sistema (data struktures)
- software za kontrolo nad celotnim sistemom (executive services)
- software za sprejemanje in izvajanje sistemskih ukazov (directive services)
- algoritmi za servisiranje prekinitev (interupt)
- sistemski odlagalnik (stack)

DELTA/M operacijski sistem ima poseben program za vzdrževanje strukturirane organizacije datotek na magnetnih medijih (diski, trakovi, diskete, ..), imenovan FILES-11. Nadzira pomnenje in upravljanje nizov (datotek) in vzdržuje hierarhično strukturo, kar omogoča zelo hiter pristop k posameznemu nizu.

Servisni programi poenostavljajo delo s sistemom na večih področjih:

- manipulacija z nizi (prepisovanje, brisanje,listanje spiskov nizov, ..)

- prevajanje, asembliranje...
- za razvoj aplikativnih programov, vzdrževanje bibliotek,....
- sistemsko vzdrževanje, odkrivanje napak...

1.2.4 ZNAČILNOSTI OPERACIJSKEGA SISTEMA DELTA/M

Je večuporabniški operacijski sistem, namenjen predvsem za interaktivno delo. Uporabnikom, ki istočasno delajo na sistemu, zagotavlja resurse (čas procesorja, vhodno/izhodne enote in pomnilnik) in več vrst zaščite (zaščita z gesli, zaščita programov, direktorijev, datotek in enot).

1.2.4.1 Multiprogramiranje v realnem času

Multiprogramiranje v realnem času zahteva natančno organizirano interakcijo naslednjih glavnih elementov:

- Izvršilnega programa. Izvršilni program predstavlja jedro operacijskega sistema in usmerja izvajanje programov.
- Programov. Program je zaporedje ukazov, ki pri manipuliranju s podatki usmerjajo računalnik tako, da doseže zastavljeni cilj.
- Pomnilnika. Pomnilnik je spominski medij v katerem se programi dejansko nahajajo in izvajajo.

1.2.5 ORGANIZACIJA POMNILNIKA

Vsaka naloga (task) se izvaja v vnaprej določenem strnjenem področju pomnilnika, ki se imenuje pomnilniški del ali particija (partition). Particija ima naslednje lastnosti:

- ime
- definirano dolžino
- fiksen začetni naslov
- definiran tip

Povezava med taskom in particijo je odvisna od tega, ali sistem vključuje preslikavanje (mapping) ali ne.

1.2.5.1 Sistemi s in brez preslikavanja

Operacijski sistem DELTA/M je zasnovan tako, da lahko teče skoraj na vseh modelih 16 bitnih računalnikov DELTA. Možno je direktno naslavljanje do 32K pomnilniških besed. Če so potrebni pomnilniki večji od 32K besed je na razpolago posebna enota , ki se ime-

nuje KT11. Ta posebna strojna oprema povezuje virtualne naslove programa (od 0 do 32K) z dejanskimi lokacijami v pomnilniku, to je s fizičnimi naslovi. Fizični naslovi lahko segajo v območje od 0 do 124K besed, v posebnem primeru pa v območje od 0 do 1920K besed.

Sistem, ki vključuje enoto KT11, se imenuje sistem s preslikavanjem ali mapiran sistem, tisti,ki te enote ne vključuje pa sistem brez preslikavanja ali nemapirani sistem.

S tem je pogojen tudi način kreiranja programskih enot ali nalog (tasks). Preden se namreč nek preveden program lahko izvaja, ga mora poseben program (Task Builder) obdelati v programsko enoto, imenovano 'naloga'ali task. Če je sistem nemapiran mora uporabnik točno specificirati naslov tiste particije, v katerem se bo task izvajal. Omenjeni task se potem ne bo mogel izvajati drugje, kot na specificiranih lokacijah.

Pri mapiranih sistemih pa ima vsak task virtualni začetni naslov 0. Enota KT11 virtualne naslove naloge poveže z dejanskimi fizičnimi naslovi, na katerih se ta naloga nahaja. Ta postopek, ki je interne narave in uporabnika ne zadeva, omogoča, da se naloge izvajajo v katerimkoli dovolj veliki particiji.

1.2.5.2 Tipi particij

Particije lahko razdelimo glede na to, kdo jih vodi na:

- sistemske (dinamične) particije
- uporabniške particije

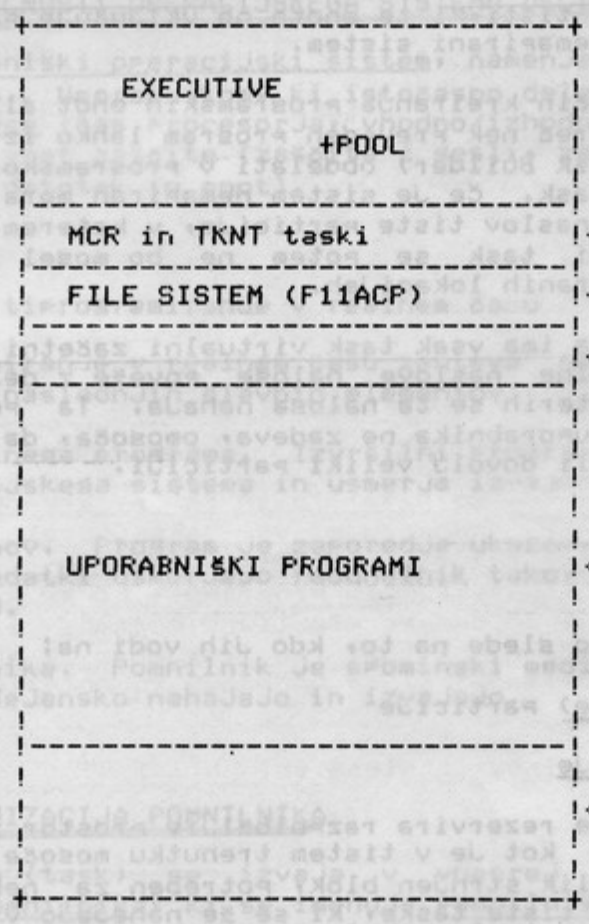
V prvem primeru executive rezervira razpoložljiv prostor za namestitev tolikih taskov kot je v tistem trenutku mogoče. Da se lahko zagotovi dovolj velik strnjen blok, potreben za nek task, sistem včasih premešča tiste taske, ki se že nahajajo v pomnilniku. Ta način dodeljevanja particij posameznim taskom je mogoč samo pri mapiranih sistemih.

V drugem primeru, ko gre za uporabniško particijo,lahko uporabnik določa v katerem delu pomnilnika se bo neka naloga izvajala. Vendar se bo v tistem trenutku tam lahko izvajala samo tista naloga. Ta tip particij je mogoč tako pri mapiranih kot pri nemapiranih sistemih.

1.2.5.3 Particije in podparticije

Uporabniško particijo lahko razdelimo naprej na najmanj sedem podparticij, ki se ne prekrivajo. Kot matična particija, lahko tudi podparticija istočasno vsebuje samo en task. Ker podparticije zasedajo isti fizični pomnilnik kot matična particija, se taski

ne morejo istočasno nahajati v njej in eni ali več podparticijah. Ker pa vsaka podparticija lahko vsebuje svoj task, potencialno lahko vzporedno teče sedem nalog, seveda če je bila matična particija predhodno izpraznjena. Namen razdeljevanja pomnilnika na podparticije je izrabiti večja pomnilniška področja nemapiranih sistemov.



sl 14. primer razdelitve pomnilnika na particije

1.2.6 VLOGA IZVRŠILNEGA PROGRAMA

Centralna procesna enota (CPU) je tisti del računalnika, ki izvršuje ukaze. Kontrolo nad CPU-jem ima v danem trenutku lahko samo en task. Multiprogramiranje je mogoče, ker izvajanje naloge vključuje se nekaj več kot samo uporabo CPU. Neka časovno zahtevna naloga npr.sproži določen proces, nato pa čaka, da se bo zaključil. Medtem ni nujno, da uporablja CPU, zato executive takrat lahko prenese kontrolo nad CPU-jem na drugi task. Multiprogramiranje omogoča boljši izkoristek CPU-ja in ostalih komponent računalnika.

Pri sistemu DELTA/M executive koordinira izvajanje vseh taskov, ki se nahajajo v pomnilniku, z namenom, da se doseže efektivna uporaba sistemskih virov in hiter odzivni čas.

Na osnovi stanja taska in prioritete taska izvršilni program dodeljuje procesor in ostale resurse sistema taskom. Pri koordinaciji dela večih taskov po potrebi uporablja sledeče postopke:

- reorganizira particije (samo sistemske)
- umika in ponovno nalaga taskov (checkpointing)
- umika z dodeljevanjem negativne prioritete (swapping)
- rotira taske z isto prioritet (round-robin scheduling)
- upotablja sistem značilnih dogodkov (significant events)

1.2.6.1 Stanje taska

Ko uporabnik z ukazom prek terminala instalira task, sistem zabeleži parametre te naloge v posebno sistemsko tabelo, ki se imenuje sistemski seznam nalog (STD - System Task Directory). Zabeleženi parametri vključujejo ime in velikost naloge, naslov na disku, kjer se task nahaja in ime particije, v kateri se bo naloga izvajala.

Task je instaliran takrat, kadar se nahaja v STD. Instaliran task pa ni niti prisoten v pomnilniku, niti se ne poteguje za sistemske vire. Executive ga ima registriranega kot mirujoči, dokler od tekočega taska ali z ukazom prek terminala ne dobi zahteve za njegovo aktiviranje. Executive torej razpoznava dve stanji taskov:

- Stanje mirovanja. Mirujoči task je tisti, ki je instaliran in se nahaja v STD, ni pa še bilo zahteve za njegovo izvajanje.

- Stanje aktivnosti. Task je aktiven, kadar je instaliran in se tudi izvaja. Aktiven ostane dokler ga ne zaključimo ali izločimo, ko se vrne v stanje mirovanja.

Aktiven task pa ima lahko še dve podstanji in sicer je lahko pripravljen za izvajanje ali blokiran.

- Taski pripravljeni za izvajanje,se na osnovi prioritete potegujejo za CPU. Dobi jo tista task, ki ima najvišjo prioriteto in tako postane tekoči task (task v izvajanju).

- Blokiran task se ne more potegovati za CPU. To je lahko iz sinhronizacijskih razlogov, ali pa zato, ker želeni vir ni razpoložljiv.

Razlikovanje med mirujočimi in aktivnimi taski je pri sistemu, ki dela v realnem času, zelo pomembno. Mirujoči task potrebuje malo spomina, kadar pa je potrebno ga executive hitro in učinkovito lahko vključi v aktivno posegan po sistemskih virih. število

mirujočih taskov je lahko, in ponavadi tudi je, večje od števila aktivnih.

Ko executive dobi zahtevo za aktiviranje mirujočega taska, stori naslednje:

- rezervira zahtevane pomnilniške vire (pomnilniški prostor);
- task prenese v pomnilnik, če je v njegovem pomnilniškem delu prostor
- aktivira task, da se z drugimi, ki se tam nahajajo, potegujejo za sistemske vire;

Če je particija, v katerem je task instaliran, zaseden in se nobenega drugega taska ne da umakniti, se ta task skupaj z drugimi aktivnimi na osnovi prioritete postavi v vrsto. Tam čaka, da bo zanj na voljo prostor v njegovi particiji.

1.2.6.2 Prioriteta

Aktivni taski se potegujejo za sistemske vire na osnovi prioritete in slede na razpoložljivost virov. Prioriteta taska je določena s številko, ki jo task dobi ob tvorjenju taska ali izvajanju. Ta številka je v območju 1 do 250 decimalno. Višja številka pomeni višjo prioriteto. Task z najvišjo prioriteto, ki ima dostop do vseh potrebnih virov, ima tudi kontrolo nad CPU. Kadar je task blokiran (ko npr. čaka, da se zaključi V/I operacija), executive poišče drug task, ki potrebuje CPU. Izbere tistega, ki ima najvišjo prioriteto in dostop do vseh potrebnih virov.

Pri operacijskem sistemu DELTA/M, ki sočasno izvaja časovno zahtevne aplikacije in manj zahtevna opravila, so časovno zahtevnim aplikacijam dodeljene višje prioritetne številke. Takšna ureditev zagotavlja, da ti taski dobijo na razpolago centralno procesno enoto pred ostalimi.

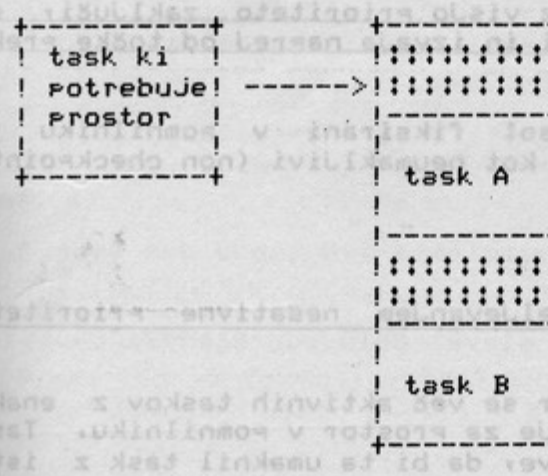
1.2.6.3 Reorganizacija particije

V primeru, ko neka sistemsko kontrolirana particija ni povsem zasedena s programi, executive poskuša vnesti task, ki čaka na vnos. Prostor za task mora biti povezan. Če je prostora sicer dovolj, a nobeden od praznih povezanih prostorov v pomnilniku ni dovolj velik za čakajoči task, executive izvrši reorganizacijo particije. Aktivira poseben task 'shuffler', ki zbere 'luknje' v particiji v povezan prazen prostor. Delo opravi v dveh korakih:

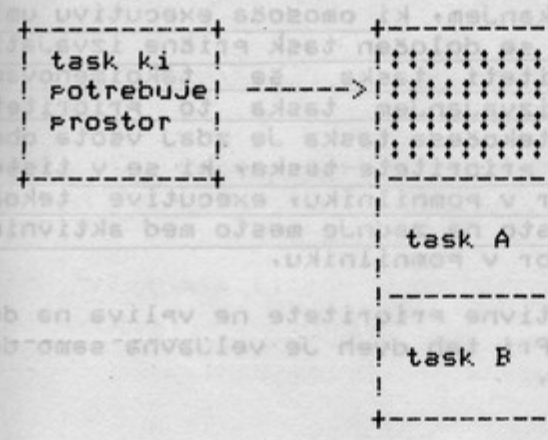
Prvi korak: premakne vse taske na začetek praznega prostora, ki je pred posameznim taskom. Umakne tudi vse taske, ki čakajo vnos s terminala. Stem doseže, da je ves prazen prostor v particiji v

enem kosu. Če tako dobljen prostor ni dovolj velik, nadaljuje z drugim korakom.

Drugi korak: Preveri, ali bi z umikanjem taskov z nižjo prioriteto dobil dovolj prostora. Če je odgovor pozitiven, pokliče program za umikanje taskov.



sistuacija pred reorganizacijo particije



situacija po reorganizaciji particije

sl 15. primer reorganizacije particije

Nekaterih taskov shuffler ne more premikati. In sicer:taski, ki uporabljajo laboratorijske in industrijske V/I enote, v pomnilniku fiksirane taske, dinamična področja itd.

1.2.6.4 Umikanje in ponovno nalaganje nalog (Checkpointing)

Včasih se kakšna naloga ne more potegovati za procesor, ker je njena particija zasedena. Če ta particija vsebuje task, ki ima nižjo prioriteto in je umakljiv, potem executive ta task prenese iz pomnilnika na disk in s tem naredi prostor tasku z višjo prioriteto. Ko se izvajanje taska z višjo prioriteto zaključi, se prej umaknjen task ponovno naloži in izvaja naprej od točke prekinitve.

Umakniti se ne morejo taski, ki so: fiksirani v pomnilniku (s posebnim ukazom), so narejeni kot neumakljivi (non checkpointable), imajo V/I operacijo.

1.2.6.4.1 Umikanje nalog z dodeljevanjem negativne prioritete (Swapping)

Kot rečeno problem nastane, kadar se več aktivnih taskov z enako ali podobno prioriteto potegujejo za prostor v pomnilniku. Task sam ne more vplivati na executive, da bi ta umaknil task z isto ali višjo prioriteto. Take okoliščine lahko nalogam z enako ali nižjo prioriteto onemogočijo dostop do pomnilnika.

Problem se reši z neke vrste umikanjem, ki omogoča executivu umik nalog s podobno prioriteto. Ko se določen task prične izvajati, executive doda običajni prioriteti taska še takoimenovano 'swapping' prioriteto. Med izvajanjem taska to prioriteto zmanjšuje. Dejanska prioriteta tekočega taska je zdaj vsota obeh prioritet. Če je ta manjša od prioritete taska, ki se v tistem trenutku tudi potegujejo za prostor v pomnilniku, executive tekoči task umakne. Postavi ga v vrsto na zadnje mesto med aktivnimi taski, ki se potegujejo za prostor v pomnilniku.

Ta postopek z dodeljevanjem negativne prioritete ne vpliva na dodeljevanje CPU in V/I enot. Pri teh dveh je veljavna samo dejanska prioriteta tekočega taska.

1.2.6.4.2 Rotiranje nalog z isto prioriteto (Round-robin Scheduling)

Executive dodeljuje CPU na osnovi prioritete. Kadar ima več aktivnih taskov pripravljenih za izvajanje enako prioriteto takrat se CPU dodeli na principu Round-robin Scheduling - rotiranja nalog z isto prioriteto. Executive najpogosteje dodeli CPU tisti nalogi, ki se v sistemskem seznamu nalog (STD) nahaja na prvem mestu. V sistemskem seznamu so naloge z isto prioriteto navadno razvrščene po vrstnem redu instaliranja. Znotraj STD algoritem razvrščanja periodično rotira naloge z isto prioriteto tako, da vsaka naloga v določenem trenutku pride na prvo mesto. Na ta način je dodeljevanje CPU bolj pravično.

1.2.7 ZAKLJUČEK

S tem smo spoznali osnovne mehanizme delovanja operacijskega sistema DELTA/M. Strukturo FILES-11 in nekatere pomožne programe bomo spoznali kasneje.

V A J E

Za vajo odgovori na sledeča vprašanja:

1. Kateri so osnovni funkcionalni deli računalnika?

2. Kaj povezuje osnovne funkcijske ~~funkcijske~~ dele med seboj?

3. Na kakšnem principu komunicirata med seboj dve enoti?

4. Na kakšen način (master-slave) delujejo sledeče enote:

DISK	S	M	M
PROCESOR	M	M	S
SPOMIN	S	S	

5. Po kakšnih lastnostih se razlikujejo med seboj procesorji?

6. Koliko bitni so procesorji sistemov DELTA/M?

7. Na katere funkcionalne dele se deli procesor pri DELTA sistemih?

8. Opišite organizacijo pomnilnika (central memory)

9. Definirajte pojem "beseda" !

10. Kakšne adrese imajo zgornji zlogi; kakšne adrese imajo spodnji zlogi?

11. Ko je instaliran 18-bitni memory management enota, je maksimalna velikost pomnilnika:
    - 104 K besed
    - 128 K besed
    - 124 K besed
    - 32 K besed

12. Kako delimo V/I enote?

13. Naštejte vsaj tri različne V/I enote.

14. Kaj je operacijski sistem (OS)

15. Naštejte čim več funkcij splošnega operacijskega sistema

16. Naštejte najpogostejše tipe OS in opišite njihove značilnosti

17. Kakšnega tipa je OS DELTA/M?

18. Naštejte glavne komponente OS DELTA/M

19. a) Kaj je izvršilni sistem (EXECUTIVE)?
    b) Kakšne naloge opravlja EXECUTIVE?
    c) Katere so osnovne komponente EXECUTIVA?

20. Na katerih področjih poenostavljajo servisni programi (UTILITIES) delo s sistemom?

21. Definirajte pojem TASK

22. a) Kaj je PARTICIJA
    b) Naštejte lastnosti particije

23. Pojasnite razliko med uporabniško particijo in sistemsko kontrolirano particijo

24. Kakšna stanja taskov poznate?

25. Pojasnite pojem prioriteta taska

26. Čemu služi sistemski seznam taskov (STD)?

27. Kakšna sredstva uporablja EXECUTIVE, da se doseže efektivna uporaba sistemskih virov in hiter odzivni čas?

28. Opišite, kaj dela SHUFFLER

29. Čemu služi umikanje in ponovno nalaganje taskov (CHECK-POINTING)?

- taski, ki se izvajajo v isti particiji se lahko zamenjajo

30. Kaj naredi EXECUTIVE, če se hkrati poteguje za CPU več taskov z enako ali podobno prioriteto?

swapping - ko se taski začne izvajati, dobijo swapping prioriteto, med izvajanjem ta prioriteto zmanjšuje. Začasno tekoči task je na postor v pomnilniškem terminu celo čas z višjo prioriteto, trenutni pa čaka

POGLAVJE2

TERMINAL

S sistemom DELTA/M komuniciramo preko terminala. Poseben nabor ukazov, imenovan Monitor Consol Routine (MCR), služi kot vmesnik med terminalom oz. uporabnikom na tem terminalu ter operacijskim sistemom DELTA/M.

2.1 RABA TERMINALA

Terminal je sestavljen iz tastature in zaslona ali pomičnega papirja. Tastatura je podobna tisti na pisalnem stroju. Poleg običajnih tipk, ki jih ima pisalni stroj, pa tastatura vključuje še posebne tipke za različne funkcije računalnika. Te tipke niso na vseh termnalih na istem mestu. Z to prevelimo razporeditev tipk vsakič, ko delamo z računalnikom, ki ga še ne poznamo.

* funkcijske tipke računalnika
  Nekatere funkcijske tipke, ki nastopajo na tastaturi so opisane v tabeli 2-1. Za specifično informacijo o programski uporabi funkcijskih tipk preverite dokumentacijo za sistemski program, ki ga uporabljte.

* kontrolni znaki
  Kontrolni znak lahko vnesemo s pritiskom na ustrezno tipko, medtem ko pritiskamo na tipko <CTRL>. Enako reagira sistem s pritiskom na strešico (^) nato pa na tipko (primer:<^U>) Nekateri kontrolni znaki se zaradi njihove funkcije ne izpišejo na terminal.

Tabela 2-1 Posebne tipke terminala

funkcijska tipka	opis
<CR>ali<RETURN>	Konča vnos vrste in prenese cursor na 1. mesto v novi vrsti(cursor je utripajoč znak na mestu pisanja) na kratko: če hočete nadaljevati vnos v novi vrsti, pritisnite <CR>ali<RETURN>
<CTRL/n>	Kombinacija tipke <CTRL> in neke ustrezne črke izvrši množico različnih funcij. Vsaka veljavna kombinacija se imenuje kontrolni znak, predstavljen z "n", kjerje "n" spremenljiva črka
<RUBOUT>ali<DELETE>	Izbriše zadnji znak v tekoči vrsti. če tipko pritisnemo večkrat, postopoma briše znake proti levi. Zaslonski terminal odstrani vsak izbrisani znak z zaslona in premakne mesto pisanja nazaj. Tiskalni terminal pa izpiše poševno črto (\), nato vsak znak, ki ste ga izbrisali, spet poševno črto (\), nato pa šele prvi pravilni znak. primer: NAPALE\ELE\KA
<TAB>	Premakne cursor na naslednji TAB-stop. TAB-stop se nahaja na vsaki osmi poziciji v vrsti. Z drugimi besedami, prvi TAB-stop je znak(pozicija) st.9.

Pri uporabi različnih sistemskih programov ti kontrolni znaki ne delujejo vedno tako kot je opisano. Na primer:nekateri programi ne spoznajo <CTRL/Z> kot izhod iz programa(tabela 2-2).Zaradi tega preverimo pomen kontrolnih tipk pri posameznih sistemskih programih, preden jih uporabljamo.

Tabela 2-2 kontrolni znaki terminala

znak	opis
<CTRL/C>	Pokliče monitor, ki razlaga ukaze operacijskemu sistemu. Na znak <CTRL/C> sistem odgovori z monitorskim znakom pripravljenosti: MCR> Ta signal označuje, da je sistem pripravljen sprejeti vnos s terminala.
<CTRL/O>	Izmenično prekine in vključi izpis na na terminal. Če nam program izpisuje nezaželene podatke, pritisnemo <CTRL/O> za ustavljanje izpisa podatkov! Če ne pritisnemo ponovno <CTRL/O>, bo sistem zaustavil ves izpis in nato izpisal znak pripravljenosti (>) in nam s tem dal vedeti, da je iznos končan.
<CTRL/S>in<CTRL/Q>	<CTRL/S> ustavi izpis na terminal dokler ne pritisnemo <CTRL/Q> in izpis se bo nadaljeval na mestu, kjer smo ga prekinili s CTRL/S.
<CTRL/R>	Premakne cursor v novo vrsto (kot <CR>) in izpiše tekočo vrsto na terminal brez eventuelnih popravkov. primer: NAPALEJELJKA <CTRL/R> NAPAKA
<CTRL/U>	Izbriše celotno tekočo vrsto. To nam omogoča, da na novo vnesemo celo vrsto, kadar bi bili posamezni popravki nepraktični. Zapomnimo pa si, da moramo, pritisniti <CTRL/U> pred <CR>, če hočemo izbrisati vrsto.
<CTRL/Z>	Prekine izvajanje večine sistemskih in uporabniških programov in vrne kontrolo monitorju.

Zapomnimo si razliko med parom ukazov <CTRL/O>-<CTRL/O> in <CTRL/S>-<CTRL/Q>!

- ko drugič pritisnemo <CTRL/O> se nadaljuje izpis od mesta, do katerega je med ustavitjijo "prebral" računalnik.
  primer:pri izpisu naravnih števil od 1-100 smo ustavili pri številki 42. Med ustavitvijo pa je računalnik prišel do št.52 in zato nadaljuje izpis podatkov od št.52.
- ko pa pritisnete <CTRL/Q> po <CTRL/S>, se izpis podatkov nadaljuje od tam, kjer smo ga ustavili z <CTRL/S>.
  primer:ustavili smo pri št.42. Izpis se nadaljuje od št.42.

2.1.1 ODZIVNI ZNAKI

Ko sedemo za terminal naredimo sledeče:

- preverimo ali je terminal vključen
- preverimo ali je terminal postavljen na "sistem" (v SET-UP-u)
- upoštevamo navodila za nastavitev terminala in povezovanje z računalnikom (direktno ali preko modema).

Če je terminal vključen in pravilno nastavljen se, ko vključimo terminal (stikalo na 1 ali DC ON) in pritisnemo na tipko <RETURN>, pojavi na levi strani zaslona ti. običajni (privzeti) odzivni znak ">" kar kaže na to, da je terminal pripravljen za delo in da računalnik dela.

Poznamo tri vrste odzivnih znakov:

1. privzeti (običajni) odzivni znak
   Privzeti odzivni znak (> v prvi koloni vrste) označuje, da lahko vnesemo nek MCR ukaz ali poženemo kak pomožni program. Vnešeno vrstico bo prevzel in interpretiral MCR task.

2. task odzivni znak (TSK>)
   Odzivni znak taska označuje, da lahko vnesemo ukazno vrstico nekemu tasku. Task bo to vrsto interpretiral, potem pa vrnil svoj odzivni znak.Privzeti prompt dobimo, če namesto ukaza za task pritisnemo CTRL/Z.

3. posebni MCR odzivni znak
   MCR> odzivni znak dobimo, če pritisnemo CTRL/C. Ko imamo privzeti odzivni znak, nimamo potrebe, da bi priklicali MCR> odzivni znak, saj bo tudi brez tega ukaze interpretiral MCR task. Če pa želimo izdati kak MCR ukaz, ko je naš terminal vezal kateri od taskov, pritisnemo CTRL/C in dobimo MCR>. Izdamo lahko en MCR ukaz. Ko se bo ta ukaz izvedel, bo nadaljeval svoje delo task, ki smo ga prekinili.

2.1.2 NEKATERA PRAVILA

Monitorski ukaz lahko izdamo, če imamo v tekoči vrsti > ali MCR> odzivni znak. To naredimo tako, da vnesemo vrstico, ki jo bo razumel task Monitor Console Routine. Ukazna vrstica se sestoji iz treh delov: ukaza, parametra, neobvezne kretnice ter znaka <CR>.

ukaz parameter/kretnica<CR>

ukaz se sestoji iz treh ali več črk in zaključeno s praznim mestom. Vsak ukaz natanko določa monitorsko funkcijo. Monitor prebere samo prve tri črke. Dodatne črke lahko pomagajo uporabniku prepoznati ukaz. (Edina izjema tega tričrkovnega pravila je ukaz HELP.Natipkati moramo celoten ukaz, da ga lahko ločimo od monitorskega ukaza HELLO.) Nekaj primerov monitorskih ukazov:

ABOrt	ustavi tekoči program
DMOunt	logično odjavi enoto
UFD	ustvari seznam uporabnikovih datotek

parameter Natančneje določi objekt ukaza, ki je navadno program ali enota. Eno ali več praznih mest mora ločiti parameter od imena ukaza ali parametre med seboj. primer: ukazu ABORT, morate dodati tudi ime tekočega programa. Naslednji ukaz bo pognal program PRIMER:

RUN PRIMER

Naslednji ukaz pa bo prekinil izvajanje program PRIMER:

ABO PRIMER

kretnica Spremeni dejansko funkcijo ukaza ali pa parameter ukaza. Kadar kretnica spremeni funkcijo ukaza, moramo pustiti prazno mesto med imenom ukaza in kretnico. Primer:

ime-ukaza /kretnica[=vrednost]

Kadar kretnica spremeni parameter ukaza, sledi kretnica neposredno parametru. Nekatere kretnice so določene z dodatnim številčnim argumentom, ki je označen kot /kretnica=vrednost. Primer;

parameter/kretnica=vrednost

Prav tako prazna mesta niso potrebna med posameznimi kretnicami.

parameter/kretnica/kretnica
ukaz /kretnica/kretnica

Primer:

SET /CRT=TI:

Če želimo poznati nek pomožni program, lahko naredimo to na dva načina. Če smo se odločili da naredimo z njim več operacij, pokličemo njegov odzivni znak in ko se pojavi vnesemo izza njega ukazno vrstico za ta pomožni program. Ko želimo končati, namesto ukazne vrstic vnesemo CTRL/Z.

Primer:

```
>PIP<CR>
PIP>ukazna vrstica<CR>
PIP>ukazna vrstica<CR>
    .
    .
PIP>ukazna vrstica<CR>
PIP><CTRL/Z>
>
```

Ako smo se odločili izvesti samo eno operacijo, navedemo samo ime pomožnega programa, presledek in ukazno vrstico. Primer:

```
>PIP ukazna vrstica<CR>
```

V tem priročniku se bomo držali sledečih pravil pri prikazovanju zapisa ukazov:

1. Obvezne črke ukaza bomo pisaki z velikimi črkami (običajno prvi trije znaki), ostali del ukaza pa z malimi črkami. (Primer: ABOrt)

2. Z malimi črkami bomo pisali parametre in spremenljivke, ki zavzamejo določeno vrednost šele, ko izdamo konkretni ukaz. (Primer: enota: pomeni da bomo na to mesto vstavili ime enote, na katero se ukaz nanaša.)

3. Parametre in kretnice, ki so možni a niso nujni, bomo pisali med oglatima oklepajema ([]). Izjema je UIC, kjer so oglati oklepaji del sintekse ([g,č]). (Primer: [/PMD=vrednost] kretnica PMD ni obvezen je pa možen.)

4. določene kretnice lahko zavzamejo tudi numerične vrednosti. Te vrednosti so desetiške, če so s piko na koncu in osmiške, če so brez pike na koncu.

Če je sintaksa ukaza napačna ali se zaradi kakega drugega vzroka ne more izvesti, sistem izpiše sporočilo na teminal, s katerega smo ukaz izdali. Obširnejšo razlago sporočil lahko najdete v MCR Operations Manual.

2.2 PRIJAVLJANJE, ODJAVLJANJE IN POMOČ

DELTA/M operacijski sistem je večuporabniški, zato ima vgrajene zaščitne mehanizme, ki preprečujejo, da bi nepooblaščene osebe uporabljale računalnik. Da bi sploh lahko delali na sistemu, se moramo najprej prijaviti - predstaviti. To naredomo z MCR ukazom HELLo. Uporabniki na sistemih DELTA/M so razdeljeni na skupine in vsaka skupina lahko ima enega ali več članov. Da bi se lahko prijavili, moramo poznati številko skupine in številko člana ter posebno geslo.

2.2.1 PRIJAVLJANJE - HELLO

Z ukazom Hello se uporabnik na terminalu prijavi in pristopi sistemu. Pred prijavo sistem zavrne vse monitorske (MCR) ukaze razen Hello in Help. Parametri za ukaz Hello so priimek ali uporabniška koda (UIC) in geslo. Sistem testira vnesene parametre glede na datoteko v kateri se nahajajo veljavna gesla in kode. Če geslo ustreza imenu ali uporabniški kodi, sistem naredi sledeče:

- Postavi, glede na uporabnikovo kodo terminal za privilegiran ali neprivilegiran (privilegirani so uporabniki s številko skupine manjšo ali enako 10(8).)

- Priredi pseudo ime SYO: na uporabnikov sistemski disk (to je disk na katerem so uporabnikove datoteke).

- Določi prijavno uporabnikovo kodo, to je začetna privzeta koda.

- Zapiše trenutni čas, datum in ostale informacije o uporabniku v sistemsko datoteko.

- Prikaže identifikacijo sistema in čas ter datum, ko se je uporabnik prijavil na terminalu.

- Prikaže (v določenih primerih) vsebino datoteke LB:[1,2]LOGIN.TXT. (izpis lahko zaustavite s CTRL/O).

- Preda v izvajanje monitorju (MCR) datoteko SY: [prijavna koda] LOGIN.CMD, če ta obstoja.

Oblike ukaza:

V nadaljevanju pomenijo zaviti oklepaji, da uporabnik vnese enega od navedenih parametrov.

```
>HELlo { UIC     } / geslo <CR>
       { priimek }

>HELlo <CR>
```

UIC ali priimek: {UIC / priimek} <CR>

GESLO: geslo <CR>

>HELlo {UIC / priimek} <CR>

GESLO: geslo <CR>

kjer je:

UIC To je koda za identifikacijo uporabnika. Ukaz dovoljuje naslednje štiri oblike specifikacije:

g,č
[g,č]
g/č
[g/č]

če ne želite izpis vsebine datoteke LB:[1,2]LOGIN.TXT uporabite obliki g/č ali [g/č]
Spremenljivki g in č sta oktalni števili od 1 do 377 in predstavljata številko grupe in člana.

priimek Priimek uporabnika se lahko uporabi namesto kode uporabnika.

geslo Alfanumerično geslo. Če vnašate geslo kot odgovor na vprašanje GESLO: sistem na terminalu ne prikaže vtipkane znake.

če je bil ukaz HELLO uspešen prikaže sistem naslednje sporočilo:

DELTA-M V1.2 BL26 SISTEM

pozdrav
dd-mmmm-ll hh:mm vključen na terminal TTn:

prijavni tekst (datoteka LB:[1,2]LOGIN.TXT)

kjer je:

pozdrav V odvisnosti od časa se izpiše eno od naslednjih sporočil:

DOBRO JUTRO
DOBER DAN
DOBER VEČER

dd-mmm-yy trenutni datum

hh:mm čas, ko se je uporabnik prijavil sistemu.

TTn: je terminal, na katerem smo se prijavili. n je

zaporedna številka (osmiška) terminala v konfiguraciji našega računalnika. TT označuje, da je enota terminal. V veliko primerih lahko namesto imena lastnega terminala uporabljamo ime TI: (terminal input), kar označuje terminal, s katerega je vnešen ukaz.

Ko smo prijavljeni, smemo izvajati operacije v okviru svojih pooblastil. Če smo se prijavili z neprivilegirano šifro, dobi naš terminal status "neprivilegiran" in lahko izdajamo samo neprivilegirane MCR ukaze in startamo neprivilegirane programe.

2.2.2 ODJAVLJANJE - BYE

Z ukazom BYE se uporabnik odjavi sistemu. Sistem izpiše odjavno sporočilo na CO:, prikaze na terminalu s katerega je ukaz izdan odjavni tekst, prekine vse uporabnikove neprivilisirane programe, dismountira enote, ki jih je uporabnik montiral (ko se odjavi privilisirani uporabnik, ostanejo obče enote montirane) in prekine dodelitev uporabnikovih zasebnih enot (če so).

Oblika ukaza:

```
>BYE
```

Primer:

```
>BYE
>
NASVIDENJE
10-JAN-81 12:39 TT11:  JE IZKLJUCEN
```

2.2.3 POMOČ - HELP

Z ukazom Help se na terminalu s katerega je izdan prikaže vsebina datoteke [1,2] HELP.HLP. To je edini monitorski ukaz, ki sa lahko uporabite predno ste se prijavili sistemu. Običajno se v datoteki nahaja opis prijavljanja sistemu, vendar lahko vsebino datoteke spremenimo in prilagodimo specifičnim potrebam.

Potrebno je vtipkati vse štiri črke, da bi lahko razlikovali ukaz Help od ukaza Hello.

Splošna oblika Help ukaza je:

HELP [beseda 1] [beseda 2]...[beseda 9]

Druga oblika ukaza prikaže tekst, ki se nahaja v uporabnikovi HELP datoteki.

HELP % [beseda 1] [beseda 2]...[beseda 9]

Uporabnik mora biti prijavljen sistemu, da bi lahko uporabil HELP %. Z HELP % obliko se prikaže vsebina datoteke HELP.HLP s trenutnega direktorija.

Številke pomenijo nivo in so lahko od ena do vključno devet.

Ko izdamo ukaz HELP, dobimo osnovne informacije, kako pridemo do nadalnjih informacij in katere besede smemo uporabljati. Če izdamo sedaj ukaz HELP BESEDA, zvemo več o tej besedi in katere besede so nam na voljo znotraj uporabljene besede. Te besede so sedaj na nivoju 2. Če želimo zvedeti kaj o besedi na nivoju 2, izdamo ukaz HELP BESEDA1 BESEDA2. Če ima sistem help datotek več nivojev, lahko to nadaljujemo do nivoja 9.

Uporabnik sme sam kreirati lastne help datoteke. Kako to naredimo več ni tema našega seminarja.

2.2.4 SPOROČITI - BROADCAST

Na sistemu navadno dela več uporabnikov. Z ukazom BRO lahko pošljemo kakršno koli sporočilo na en ali več terminalov. Privilisirani uporabnik lahko pošlje sporočilo na vse sistemu prijavljene terminale. Vsi uporabniki (privilisirani in neprivilisirani) lahko pošljejo sporočilo na kateri koli terminal.

V kolikor se sporočilo v 10 sekundah ne more prikazati, javi sistem ana terminalu, s katerega je bil izdano naslednje opozorilo:

BRO -- TERMINAL IS BUSY -- ttn:

To pomeni, da je terminal ttn: zaseden in sporočilo se ni prikazalo.

V kolikor privilisirani uporabnik pošilja sporočilo na več terminalov hkrati (z VSEM: ali LOG:) se opozorilo izpiše za vsak terminal, na katerem prikaz sporočila ni bil možen.

Oblike ukaza:

BROadcast ttn: sporočilo
BROadcast @imedatoteke
BROadcast VSEM:sporočilo (privilisirana oblika)
BROadcast LOG:sporočilo (privilisirana oblika)

Pomen kratic:

ttn: Terminal kateremu je sporočilo namenjeno

sporočilo sporočilo, ki ga želimo poslati, ki je sestavljeno iz niza največ 80 ASCII znakov.

VSEM: Privilisirana opcija. To pomeni, da naj se sporočilo pošlje na vse terminale v sistemu (tu niso vključeni podrejeni terminali, kot na primer RMDEMO terminal ali virtualni terminali).

LOG: Privilisirana opcija. To pomeni da naj se sporočilo pošlje na vse sistemu prijavljene terminale (tu niso vključeni podrejeni in virtualni terminali).

Ukaz BROADCAST prikaže sporočilo na zahtevanem terminalu v naslednji obliki:

dan:mesec:leto FROM:ttnn: To:TTnn:

sporočilo

Primeri:

```
>BRO VSEM: DANES OD 10 DO 12 BO SISTEM UGASNJEN
>BRO tt20: PRIDI, GREVA NA KAVO
```

2.2.5 ČAS - TIME

TIM ukaz nam omogoča:

* naravnati tekoči dnevni čas (privilegirano)

* naravnati tekoči datum (privilegirano)

* izpisati na zaslon tekoči čas in datum

Oblika ukaza:

TIM [ura:min[:sek]] [dan-mes-leto]

ali

TIM [ura:min[:sek]] [mes1/dan/leto]

kjer je:

ura	- ura (med 0 in 23)
min	- minute (med 0 in 59)
sek	- sekunde (med 0 in 59)(ni obvezno)
mes	- ime meseca skrajšano na 3 črke
mes1	- mesec izražen s številko (med 1 in 12)
dan	- dan v mesecu (med 1 in 31)
leto	- leto (med 0 in 99) šteto od 1900.

Pomni!

* Če v ukazu ne navedemo niti časa niti datuma, sistem izpiše tekoči čas in datum.

* Če privilegirani uporabnik navede v ukazu datum in čas, naravna s tem sistemski čas in datum na vrednost, ki jo je navedel. Če navede samo datum, naravna samo datum; če navede v ukazu čas, naravna samo čas.

* Numerične vrednosti so desetiške; pika na koncu števil ni potrebna.

Primer:

```
>TIM
10:23:31 03-SEP-82

>TIM 7:12 3-SEP-82
```

Čas smo naravnali na 12 minut čez 7, 3. septembra 1982.

2.3 LASTNOSTI TERMINALA

Vsak terminal ima določene harwareske in softwareske lastnosti. Te lastnosti določajo način njegovega delovanja in obnašanja v določenih situacijah. Ker vedno delamo z računalnikom preko terminala, nas operacijski sistem mnogokrat prepoznava kot terminal oz. uporabnika na tem in tem terminalu.. Terminal je v sistemu predstavljen z imenom TTn:. "n" je (osmaško) zaporedno število terminala v sistemu. Na katerem smo se terminalu prijavili vidimo takoj, ko se prijavimo. Da nam ne bi bilo potrebno ves čas vedeti, kateri je naš terminal, smemo uporabljati pseudo ime TI:, ki označuje terminal, s katerega je bil ukaz izdan.

Ukaz SET lahko dinamično menja lastnosti terminala oz. daje informacije o njih. Neprivilisiran uporabnik lahko menja samo lastnosti svojega terminala (TI:) in svojih privatnih enot. Privilisirani uporabnik lahko menja lastnosti vsakega terminala in drugih enot. Oba pa lahko dobita informacije o enotah oz. sistemu.

Neprivilisirani uporabnik sme:

- določiti nekatere lastnosti svojega terminala
- spremeniti UIC svojemu terminalu
- izpisati informacije za vsak drug terminal oz. vse terminale

Privilisirani uporabniki imajo mnogo več možnosti. Privilisirane SET (pa tudi druge privilisirane ukaze) lahko izdajamo s privilisiranega terminala. Terminal pa je privilisiran, če smo se na njem prijavili s privilisirano šifro ali je terminal naravnan kot privilisiran.

Oblika ukaza:

SET /kretnica[=vrednost]

Vsak SET ukaz sme imeti samo eno kretnico. Pri nekaterih kretnicas s predpono NO spremenimo delovanje v negativno smer.

/BUF=enota:[dolžina]

/CRT[=TTnn:]

/ECHO[=TTnn:]

/LOWER[=TTnn:]

/SLAVE[=TTnn:]

/SPEED=TTnn:[sprejem:oddajanje]

/TERM=TTnn:[tip]

/UIC[=[uic][TTnn:]

Opis kretnic:

/BUF=enota:[dolžina]

Naravna ali izpiše dolžino medpomnilnika navedene enote. Neprivilegirani uporabnik lahko menja dolžino medpomnilnika svojega terminala TI: in svojih privatnih enot. Dolžina medpomnilnika mora biti med 0 in 255(10) za enote in med 2 in 255(10) za terminale. Če dolžine ne navedemo, dobimo na ekranu dolžino medpomnilnika navedene enote oz. terminala.
Primer:

```
>SET /BUF=TI:
BUF=TT5:80.
```

Dolžina medpomnilnika našega terminala je 80(10) znakov.

```
>SET /BUF=TI:40.
```

Dolžina medpomnilnika je sedaj 40(10) znakov.

/CRT[=TTnn:]

Če terminala ne navedemo, nam izpiše vse terminale, ki so naravnani kot CRT (Cathod Ray Tube)- zaslonski terminal. Za te terminale je značilno, da se ob pritisku na tipko DELETE utripalka pomakne za en znak nazaj in ga zbriše. Če napišemo SET /CRT=RI:, naravnamo naš terminal kot zaslonski.
Primer:

```
>SET /CRT=TI:
```

Naš terminal je sedaj zaslonski.

```
>SET /CRT
CRT=TT3:
CRT=TT4:
CRT=TT7:
```

Na zaslonu smo dobili izpisane vse terminale, ki so zaslonski.

/NOCRT[=TTnn:]

Če terminala ne navedemo, nam izpiše vse terminale, ki so naravnani kot NOCRT. Za te terminale je značilno, da se ob pritisku na tipko DELETE izpiše '\' in prvo črko, ki smo jo želeli zbrisati. Če pritisnemo tipko DELETE ponovno, izpisuje črke, ki jih brišemo v nasprotnem vrstnem redu kot so napisane. Ko prvič pritisnemo tipko različno od DELETE, ponovno izpiše '\' in znak ki smo ga pritisnili. Če napišemo SET /NOCRT=TI:, naravnamo naš terminal kot NOCRT.

Primer:

```
>SET /NOCRT=TI:
```

Naš terminal je sedaj NOCRT.

```
>SET /NOCRT
CRT=TT0:
CRT=TT1:
CRT=TT2:
CRT=TT5:
CRT=TT6:
```

Na zaslonu smo dobili izpisane vse terminale, ki niso zaslonski.

/ECHO[=TTnn:]

Če terminala ne navedemo, se na zaslon izpišejo vsi terminali, ki so naravnani ECHO; ECHO pomeni, da se vsak znak, ki ga pritisnemo na tastaturi odrazi na zaslonu. Če v ukazu navedemo naš terminal oz. TI:, se naš terminal naravna kot ECHO, če prej ni bil tako naravnan. Terminali so navadno naravnani ECHO.
Primer:

```
>SET /ECHO
ECHO=TT0:
ECHO=TT1:
ECHO=TT2:
ECHO=TT3:
ECHO=TT4:
ECHO=TT5:
ECHO=TT6:
ECHO=TT7:
```

/NOECHO[=TTnn:]

Z ukazom SET /NOECH=TI: odpravimo odzivnost na zaslonu. Pritisk na tipko se na zaslonu ne odraža če prav sistem sprejme, kar tipkamo. Sporočila sistema so vidna na zaslonu. Nepriviligirani uporabnik lahko odpravi odzivnost le na svojem terminalu. Lahko pa dobi informacijo, kateri terminali so naravnani NOECHO.
Primer:

```
>SET /NOECHO=TT5:

TT5:  smo odpravili odzivnost.

>SET /NOECHO
NOECHO=TT5:
```

/LOWER[=enota:]

Enota je lahko terminal ali printer. S to kretnico naravnamo enoto tako, da navedena enota razlikuje male in velike črke, tj. male črke ne pretvarja v velike, ko jih sprejema s tastature ali driverja. Neprivelisiran uporabnik lahko dobi informacijo, katere ne pretvarjajo male črke v velike (s SET /LOWER). Spremembe pa lahko vrši samo za TI: (s SET /LOWER=TI:)
Primer:

```
>SET /LOWER
LOWER=TT1:
LOWER=TT2:
LOWER=TT3:
LOWER=TT7:
```

/NOLOWER[=enota:]

Enota je lahko terminal ali printer. S to kretnico specificiramo, da navedena enota ne razlikuje male in velike črke, tj. male črke pretvarja v velike, ko jih sprejema s tastature ali driverja. Neprivelisiran uporabnik lahko dobi informacijo, katere enote pretvarjajo male črke v velike (s SET /NOLOWER). Spremembe pa lahko vrši samo za TI: (s SET /NOLOWER=TI:)
Primer:

```
>SET /NOLOWER
NOLOWER=TT0:
NOLOWER=TT4:
NOLOWER=TT5:
NOLOWER=TT6:
```

/SLAVE[=TTnn:]

TTnn: je terminal. Če nek terminal naravnamo ko SLAVE, lahko preko njega vnašamo le podatke na zahtevo nekega taska. Preko takega terminala ne moremo izdajati monitorskih ukazov. Če terminala ne navedemo, dobimo izpisane vse terminale v konfiguraciji računalnika, ki so naravnani SLAVE. Neprivilisirani uporabnik lahko naravnava le svoj terminal, o ostalih pa lahko dobiva le informacije.
Primer:

```
>SET /SLAVE
SLAVE=TT3:
SLAVE=TT4:
SLAVE=TT7:
```

/NOSLAVE[=TTnn:]

Ima nasprotni učinek kot kretnica SLAVE. Terminal, ki je naravnan SLAVE, naravna kot NOSLAVE in potem lahko spet izdajami monitorske ukaze preko tega terminala. Naravnanost spremenimo tako, da navedemo terminal, za katerega naj ukaz velja. /NOSLAVE=TTnn:. Če terminala ne navedemo, dobimo na

zaslonu izpisane vsa NOSLAVE terminale. Naravnanost lahko spremeni samo privilisiran uporabnik.
Primer:

```
>SET /NOSLAVE
NOSLAVE=TT0:
NOSLAVE=TT1:
NOSLAVE=TT2:
NOSLAVE=TT5:
NOSLAVE=TT6:
```

/SPEED=TTnn:[sprejem:oddajanje]

S to kretnico lahko naravnamo sprejemno in oddajno hitrost za svoj terminal, če smo ju v ukazu navedli; če hitrosti ne navedemo, dobimo na zaslonu izpisano sprejemno in oddajno hitrost terminala, ki smo ga navedli. Privilisirani uporabnik lahko menja sprejemno in oddajno hitrost za kateri koli terminal, neprivilisirani samo za svoj terminal. Sprejemna in oddajna hitrost morata biti enaki. Če smo hitrost spremenili, jo moramo na svojem terminalu tudi hardwarejsko naravnati na isto hitrost. To naredimo v SET-UP B modu. Veljavne hitrost so:50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600 in 19200 bitov/sek.
Primer:

```
SET /SPEED=TT5:
SPEED=TT5:2400:2400
```

Terminal TT5: sprejeme in oddaja s hitrostjo 2400 bit/sek.

/TERM=TTnn:[tip]

Če tip terminala ne navedemo, dobimo na zaslon izpisano, kakega tipa je navedeni terminal. Tip terminala lahko naravnamo le za svoj terminal in to tako, da navedemo, kakega tipa naj bo naš terminal; možni so sledeči tipi terminalov: VT100, VT52, LA36, LA120 in še nekateri drugi. Privilisirani uporabnik lahko naravna tip katerega koli terminala.
Za naravnavanje tipa terminala je možna tudi malo drugačna sintaksa: >SET /tip-terminala=TTnn:.
Primer:

```
>SET /VT52=TI:
```

Svoj terminal smo narevnali na tip VT52.

```
>SET /TERM=TT5:
TERM=TT5:VT100
```

Terminal TT5: je tipa VT100.

/UIC[=[uic][TTnn:]]

S SET /UIC=[g,č] spremenimo privzeto vrednost za UIC za vse operacije, kjer je UIC igra kako vlogo. S tem preidemo na drug spisek datotek na istem disku (če ta obstaja). Z ukazom SET /UIC dobimo na zaslon izpisan naš privzeti UIC. Če uic izpustimo in zapišemo SET /UIC=TTnn:, dobimo kakšen je trenutno privzeti UIC na terminalu, ki smo ga navedli.
Primer:

```
>SET /UIC=[50,3]

Na našem terminalu je privzeta vrednost za UIC [50,3].

>SET /UIC
UIC=[50,3]

Vidimo, da je naš UIC res [50,3]

>SET /UIC=TT7:
UIC=[200,10]

Na TT7:  je privzeta vrednost za UIC [200,10]
```

V A J E

1. Vključite terminal in preverite, ali računalnik dela.

2. Vaja uporabe tastature in kontrolnih znakov (razna sporočila sistema naj vas ne motijo):
   a) Vtipkajte vrstico: TO SO VAJE ZA TERMINAL
   b) Zbrišite vse znake do TO in vtipkajte: JE VAJA ZA UPORABO TASTATURE
   c) Vnesite kontrolni znak <CTRL/U>

3. Prijavite se sistemu z šifro, ki vam jo da inštruktor. Prijavite se na različne načine (vmes se morate seveda odjaviti).

4. Uporabite ukaz HELP. Uporabite HELP za MCR ukaz TIME, HELLO, SET, BYE, HELP in BRO.

5. Izpišite trenutni čas in današnji datum.

6. Preverite, kateri je vaš UIC

7. Pošljite sporočilo na svoj terminal. Pošljite sporočilo na sosedov terminal.

8. Poslejte kakšna je dolžina bufferja vašega terminala

9. a) Postavite vaš terminal na stanje NOCRT
   b) vpišite vrstico znakov (kot v vaji 2)
   c) popravljajte to vrstico z <DELETE>
   d) zaključite to vrstico tako, da se popravljena vrstica izpiše
   e) ponovno vzpostavite prvotno stanje

10. a) Postavite terminal na stanje NOECHO
    b) ponovno vzpostavite prvotno stanje

11. Poslejte s kakšno hitrostjo dela vaš terminal

12. Odjavite se sistemu, ugasnite terminal in rešujte ostale naloge samo pismeno! Odgovorite na vsak

stavek z 'DA', če je trditev pravilna in z 'NE', če je trditev nepravilna:

a) Uporabnik se mora prijaviti sistemu, če hoče uporabiti katerikoli MCR ukaz NE(HELP)

b) Sistem sam odjavi uporabnika, ko uporabnik usasne terminal NE

POGLAVJE3

DELO Z DATOTEKAMI

3.1 PODATKOVNE STRUKTURE

3.1.1 FIZIČNE IN LOGIČNE ENOTE PODATKOV

Računalnik dela z binarnimi podatki. Ti podatki so na različne načine zapisani in organizirani. Lahko jih pa tudi različno interpretiramo. Kako manipuliramo in interpretiramo podatke se imenuje upravljanje s podatki. S podatki upravljamo z programsko opremo, ki je temu namenjena. Programska oprema za upravljanje s podatki zagotavlja enostaven način dela s podatki i povezuje fizični zapis z logičnim pomenom.

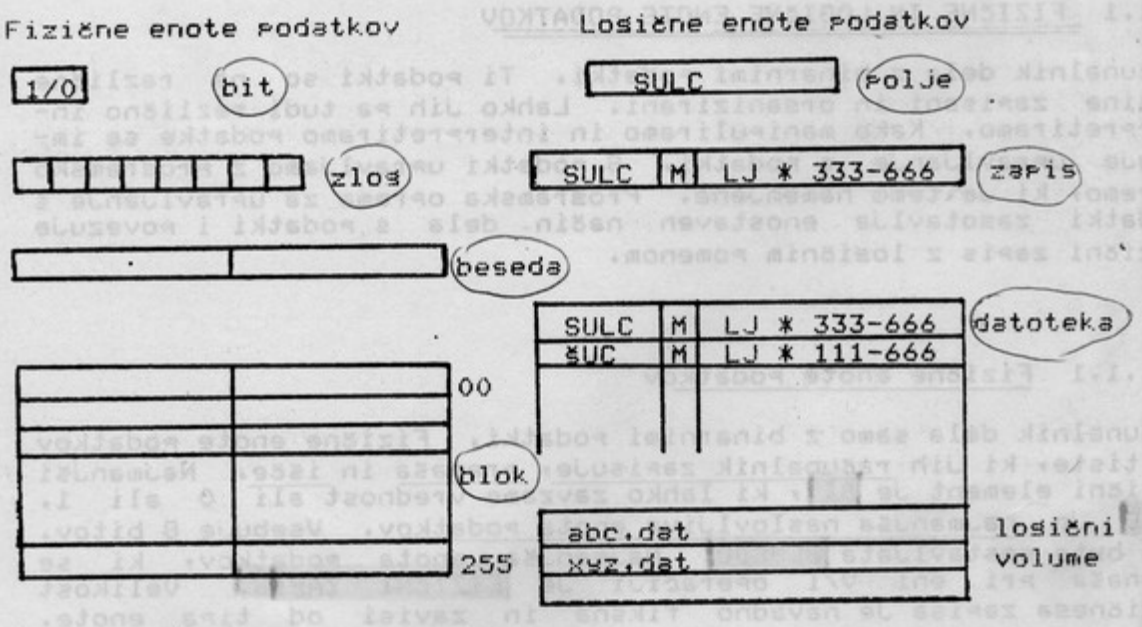
3.1.1.1 Fizične enote podatkov

Računalnik dela samo z binarnimi podatki. Fizične enote podatkov so tiste, ki jih računalnik zapisuje, prenaša in išče. Najmanjši fizični element je BIT, ki lahko zavzame vrednost ali 0 ali 1. BYTE je najmanjša naslovljiva enota podatkov. Vsebuje 8 bitov. Dva byta sestavljata BESEDO. Najmanjša enota podatkov, ki se prenaša pri eni V/I operaciji je FIZIČNI ZAPIS. Velikost fizičnega zapisa je navadno fiksna in zavisi od tipa enote. čitalec kartic lahko prečita naenkrat le 80 znakov z 80 kolonske kartice.

Pri sistemih DELTA je BLOK ime za fizični zapis na tračnih, diskovnih in disketnih enotah in znaša 512 bytov. Fizični bloki se združujejo v fizične enote ali VOLUME. Ta pojem se navadno uporablja za zamenljive medije: trakovi, diski in diskete.

3.1.1.2 Logične enote podatkov

Logične enote podatkov so elementi s katerimi manipulirajo programerji oz. aplikacijski programi. Informacije imajo logični značaj: datum, priimek, cena, itd. Logične enote podatkov so neodvisne od sistema in stvar operacijskega sistema je, da vpostavlja zvezo med fizičnimi in logičnimi enotami podatkov. POLJE je najmanjša logična enota podatkov. Podatek cena je eno polje. LOGIČNI ZAPIS je skupina polj združenih v enoto. Navadno vsebuje podatke, ki so v neki zvezi oziroma se nanašajo na nek objekt. Zančilnost logičnega zapisa določi programer. Logični zapis se lahko razteza čez več fizičnih, ali je sam v fizičnem oz več logičnih zapisov je v enem fizičnem. DATOTEKA je skupina logičnih zapisov, ki zavzema en ali več fizičnih zapisov in ki jo operacijski sistem prepoznava kot enoto. Fizični zapisi, ki pripadajo eni datoteki, so lahko razmetani ali pa povezani. LOGIČNA ENOTA je skupina datotek, ki se nahaja na eni fizični enoti. DELTA/M operacijski sistem ne podpira logičnih enot preko več fizičnih razen na traku.



3.1.2 FIZIČNA ORGANIZACIJA V/I ENOT

Vhodno/izhodne enote razdelimo na dve skupini:

- file-nestrukturirane V/I enote
- file-strukturirane V/I enote

File-nestrukturirane V/I enote lahko čitajo podatke samo v takšnem vrstnem redu, kot so fizično razvrščeni. Sem spadajo: terminali, čitalci kartic, čitalci papirnatega traku, printerji ...

File-strukturirane V/I enote omogočajo selektivno pristopanje k datotekam ali celo posameznim zapisom. Fizični zapis na file-strukturiranih enotah se imenuje fizični blok in je dolg 512 bytov. Vsak blok ima svoj fizični naslov. Naslovi gredo od 0 naprej. Med file-strukturirane enote spadajo diski, diskete, DECtrakovi, magnetni trakovi in kasete.

Skupino file-strukturiranih enot lahko razdelimo naprej na:

- enote z zaporednom pristopom
- enote z direktnim pristopom

Enote z direktnim pristopom nam dovoljujejo pisanje tako zaporedno kot tudi direktno. Enote z zaporednim pristopom dovoljujejo pisanje samo zaporedno; sem spadajo predvsem trakovi in kasete. Posamezni datoteki lahko pristopimo tako, da prečitamo vse datoteke, ki se fizično nahajajo pred želeno datoteko. Znotraj posamezne datoteke lahko prečitamo nek zapis le, če smo prej prečitali vse pred njim nahajajoče se zapise.

Enote z direktnim pristopom (diski, diskete, DECtrakovi) omogočajo direktni pristop k posameznim datotekam oz. zapisom znotraj datotek (kar pa je odvisno od logične organizacije datotek). Fizični bloki posamezne datoteke niso nujno povezani, ampak so lahko razmetani po celem mediju. Bloki, ki pripadajo določeni datoteki imajo znotraj datoteke prirejene posebne naslove - virtualne naslove, ki začnejo z nič. Virtualne adrese blokov datoteke ustrezajo logičnemu vrstnem redu. Ko sistem pristopa k datoteki, poveže virtualne naslove z fizičnimi. Zvezo med fizičnimi in virtualnimi naslovi vzdržuje poseben del operacijskega sistema.

3.1.3 STRUKTURA FILES-11

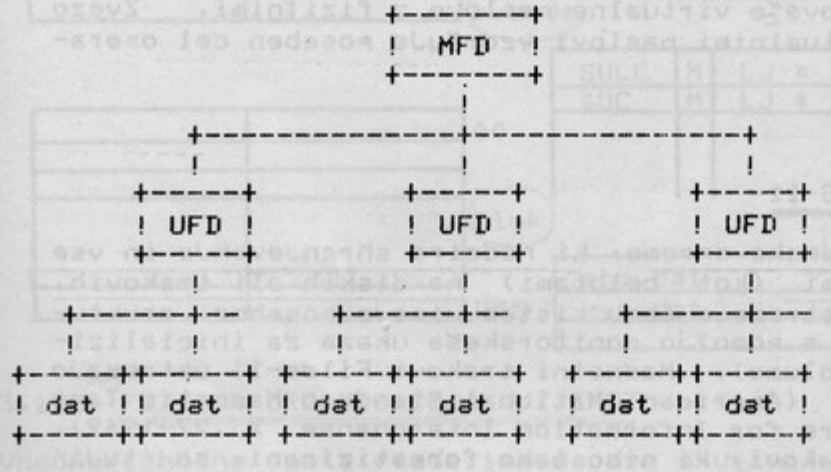
Files-11 je del programske opreme, ki nadzira shranjevanje in vse postopke z datotekami (kot celotami) na diskih ali trakovih. Magnetni mediji, ki ustrezajo temu sistemu imajo posebno strukturo, ki jo vspostavimo s pomočjo monitorskega ukaza za inicializiranje (= Initialize Volume). Magnetni trakovi Files-11 ustrezajo ameriškim standardom (American National Standard Magnetic Tape Labels , File Structure for Information Interchanse X3.27-1969). Tisti diski oz. trakovi, ki niso tako formatizirani, so 'tuji' in sistem Files-11 nima direktnega pristopa do njih. Vključuje pa DELTA/M tudi pomožni program za prevajanje datotek iz formata DOS-11 ali RT-11 v format Files-11.

Ker imajo uporabniški taski dostop do tujih magnetnih medijev, lahko opravljajo V/I operacije neodvisno od sistema Files-11. To je velikrat potrebno predvsem pri časovno zahtevnih aplikacijah.

3.1.3.1 Hierarhična struktura medijev

V sistemu Files-11 je vsaka informacija zapisana v neki datoteki. Tudi informacije o datotekah so spet v neki datoteki. Datotekam pristopamo preko imena datoteke. Ker je sistem večuporabniški, obstaja za vsakega uporabnika vsaj en spisek njegovih datotek - UFD (User File Directory). Ko uporabnik kreira datoteko, sistem poskrbi, da se ime datoteke zapiše v seznam uporabnikovih datotek UFD . Istočasno, skupaj z imenom datoteke, sistem shrani tudi uporabnikovo tekočo identifikacijsko kodo UIC (User Identification Code), ki označuje lastnika omenjene datoteke. V večini primerov UFD odgovarja lastniku UIC; datoteka pa je lahko navedena tudi v seznamu uporabnikovih datotek UFD, ki ni povezan z lastnikovo kodo. Ena datoteka se lahko istočasno prikaže v več kot enem UFD.

Seznam uporabnikovih datotek je tudi sam datoteka, ki jo mora uporabnik eksplicitno kreirati s pomočjo monitorskega ukaza UFD. Uporabnik specificira UFD v formatu za UIC: [s,č]. Pri tem sta s in č osmiški števili, ki predstavljata uporabniško skupino in člansko številko. Dejansko ime seznama uporabnikovih datotek (directory) je skoncentrirano število skupine in člana, ki se konča z .DIR. Ime programa, ki ustreza npr. uporabniški identifikacijski kodi UIC [203,165], je 203165.DIR. Vsi UFD so za vsak medij navedeni v glavnem seznamu datotek tega medija MFD (Master File Directory), ki ustreza uporabniški identifikacijski kodi UIC [0,0] in se torej imenuje 000000.DIR. Na vsakem mediju je samo en MFD.



sl 19. hierarhična struktura direktorijev na disku

V MFD-ju so poleg vseh UFD-jev na tem mediju zapisana tudi imena posebnih datotek, ki jih Files-11 rabi zase. Seznami datotek, tako UFD kot MFD, vsebujejo poleg imen datotek še kazalce, ki kažejo na glave posameznih datotek. Glava datoteke vsebuje informacije o fizični lokaciji posameznih segmentov te datoteke.

3.1.4 DOLOČITELJI DATOTEKE (FILE SPECIFIERS)

Da se uporabnik lahko kakorkoli sklicuje na neko datoteko, mora v ukaznem nizu vnesti standardne določitelje te datoteke. Ti vsebujejo naslednje informacije:

- enoto, na kateri je pomnilniški medij z ustrezno datoteko
- seznam UFD v katerem se ustrezna datoteka nahaja
- ime datoteke
- okrajšavo za tip datoteke (pojasnjuje vsebino datoteke)
- število, ki pomaga razlikovati posamezne verzije iste datoteke (številka verzije)

Vse to skupaj imenujemo "polno ime datoteke". Za določene dele polnega imena datoteke obstajajo določene privzete vrednost (defaulti). To so vrednosti, ki jih operacijski sistem privzame, če ta del imena izpustimo. Polno ime datoteke ima sledečo obliko in veljajo sledeča pravila.

enota:[s,č]ime.tip;verzija

kjer je:

enota: označuje dejansko ali logično ime enote, na kateri se nahaja datoteka. Privzeta vrednost je pseudo enota SY: (uporabnikova sistemska enota)

[s,č] je direktorij (spisek datotek), v katerega je vpisana datoteka. Direktorij je označen z številko grupe in člana UIC-ja, ki mu ta direktorij pripada. Oklepaji so del sintakse. Privzeta vrednost je UIC, na katerega je uporabnik trenutno naravnan.

ime je 1 do 9 alfanumeričnih znakov, ki označujejo ime datoteke v ožjem smislu beseda. Pika vedno ločuje ime od tipa datoteke. Privzeta vrednost ne obstaja.

tip je sestavljen iz treh črk, ki označujejo vsebino datoteke. Tip je lahko tudi prazen. Obstajajo standarni tipi datotek. Tudi sami si lahko zmislimo kak tip datoteke. Podpičje (;) loči tip datoteke od verzije. Privzeti tipi so ravno standardni tipi. Nekateri taski delajo s standardnimi tipi datotek, če jih ne specificiramo.

tip	vsebina datoteke
BAS	izvorni program v basic interpreterju
B2S	BASIC-PLUS-2 izvorni program
CBL	COBOL izvorni program
CMD	posredna ukazovna datoteka

DAT	podatkovna datoteka
DIR	direktorij
FTN	FORTRAN izvorni program
LST	lista
MAC	MACRO-11 izvorni program
MAP	TKB slika alokacije spomina
MLB	macro knjižnica
OBJ	prevedena oblika programa
ODL	opis prekrivanja (overlayiranja) programa
OLB	knjižnica prevedenih modulov
SML	sistemska macro knjižnica
STB	tabela simbolov (iz TKB-ja)
SYS	sistemski task, ki ga lahko "bootamo"
TMP	začasna datoteka
TSK	izvedljivi program
TXT	tekstovna datoteka
ULB	univerzalna knjižnica

standardni tipi datotek

verzija osmiško število med 1 in 77777, ki ločuje različne verzije iste datoteke. Ko datoteko kreiramo, dobi datoteka verzijo 1. Če jo z editorjem popravljamo dobimo novo verzijo. Privzeta vrednost je zadnja verzija. V nekaterih operacijah moramo verzijo eksplicitno navesti.

3.1.4.1 ZAŠČITA DATOTEK

Vsakdo, ki hoče pristopiti k neki datoteki, mora najprej vedeti v katerem UFD se ta datoteka nahaja. To pa še ni dovolj. Uporabnik mora zadostiti tudi pogojem, ki so specificirani v posebni zaščitni maski za to datoteko. Ta maska določa tipe pristopa za posamezne skupine uporabnikov. Tipi pristopa so štirje in sicer: čitanje (R), pisanje (W), razširjanje (E) in brisanje (D) (read, write, extend, delete). Skupine uporabnikov, ki so formirane glede na njihove UIC, so naslednje:

- Sistemski taski in uporabniki s privilegiranim UIC
- Uporabniški taski in uporabniki, ki delajo pod istim UIC kot lastnik datoteke
- Taski in uporabniki katerih UIC je v isti skupini kot lastnik datoteke
- vsi drugi taski in uporabniki

Zaščita datotek je formirana tako, da za vsako skupino predpišemo, kaj sme početi z datoteko. Primer: zaščita [RWED,RWED,RWE,R] pomeni, da privilisirani uporabnik in lastnik smeta čitati, pisati, dodajati in brisati. Uporabnik iz iste

skupine kot lastnik ne sme brisati, vri ostali pa smejo le čitati.

3.1.5 POSTOPKI Z DATOTEKAMI

Datoteke lahko tvorimo, brišemo, izpišemo na ekran ali tiskalnik, spreminjamo, preimenujemo ali jim spremenimo zaščito. Te operacije izvajamo običajno s katerim od pomožnih programov ali lastnim aplikacijskim programom. Ko datoteko tvorimo, moramo napisati njeno ime in tip in datoteka se vpiše v naš direktorij - UFD. Včasih vpišemo kako datoteko v kak drug direktorij, če nam zaščita to ne preprečuje. Najpogosteje tvorimo datoteke s pomočjo editorja - pomočnega programa za pisanje tekstovnih datotek. Mnoge datoteke so rezultat naših operacij: prevajanja programov, povezovanja programov s sistemom, kopiranja datotek. Svoje datoteke lahko imamo na različnih enotah računalnika, da le imajo FILES-11 strukturo. Sistem FILES-11 ne dela razlik med datotekami, ki se nahajajo na medijih različnih tipov in zagotavlja integriteto datotek, ko se le te prenašajo z enega medija na drugega, da le imajo FILES-11 strukturo.

3.2 TVORJENJE DATOTEK IN DIREKTORIJEV

Rekli smo, da se vsaka datoteka vpiše v nek direktorij, ko jo tvorimo. Običajno nekateri direktoriji obstajajo. Vsak uporabnik na sistemu ima vsaj en direktorij. Ta direktorij se tvori istočasno z tvorbo samega UIC in gesla. Ko se prijavimo, postane ta direktorij naš privzeti direktorij. datoteke, ki jih tvorimo, se vpišejo v ta direktorij. Včasih potrebujemo neki dodatni direktorij (mogoče na drugem disku). Naredimo ga z MCR ukazom UFD. Neprivilisirani uporabnik lahko tvori novi direktorij samo na svoji privatni (allocirani) enoti.

3.2.1 KREIRANJE DIREKTORIJA

Ukaz UFD kreira novi spisek uporabnikovih datotek (UFD) na Files-11 strukturiranem mediju in ga vpiše v MFD. Da to lahko naredimo, mora biti medij inicializiran (ukaz INI) in pristavljen (ukaz MOU). Nov spisek uporabnikovih datotek lahko kreiramo kadarkoli. Neprivilisiran uporabnik lahko kreira spisek uporabnikovih datotek le na svoji privatni enoti (alocirani).

Oblika ukaza:

UFD ddn:[labela] [g,č][/kretnica(i)]

Kretnici:

/ALLOC=število-vpisov

/PRO=[sistem,lastnik,grupa,svet]

kjer je:

ddn: enota, na kateri je medij, na katerem želimo kreirati UFD. Privzata vrednost: je ni; enoto moramo vedno navesti.

labela če jo navedemo, jo sistem primerja s tisto, ki je na mediju in če se ne skladata, UFD-ja ne kreira. Privzeta vrednost: če labele ne navedemo, se primerjanje ne izvede.

[g,č] je UIC za UFD; UIC določa lastnika spiska datotek. Oklepaji so del sintakse. Privzeta vrednost: ni privzete vrednosti; UIC moramo vedno navesti.

/ALLOC /ALLOC=število-vpisov. pomeni število možnih vpisov v spisek datotek, za kar se alocira prostor. Navedeno število se zaokroži na prvi večji večkretnik števila 32(desetiško). Privzeta vrednost: 32(10)

/PRO /PRO=[sistem,lastnik,grupa,svet]. Določa pravico pristopa k spisku datotek vsem štirim kategorijam uporabnikov. Za vsako kategorijo uporabnikov določimo njihove pravice s pomočjo kode za:

R - čitanje
W - pisanje
E - dodajanje
D - brisanje

če je določena koda pri kategoriji uporabnikov izpuščena, potem ne more izvajati tistih dejavnosti, ki jih koda specificira. Privzeta vrednost za zaščito: [RWED,RWED,RWED,R]

Primer:

```
>UFD DK3:[50,7]/PRO=[RWED,RWED,RWE,R]
```

Kreirali smo UFD na DK3: za UIC [50,7] in z zaščito [RWED,RWED,RWE,R].

3.2.2 KREIRANJE DATOTEK

Tekstovno datoteko lahko kreiramo na dva načina:s PIP pomožnim programom ali pa z nekim editorjem.

S PIP-em lahko pišemo datoteko. Vrstice, ki jih vnašamo, se takoj prenašajo v datoteko na disku. Popravljanje ni možno. Naredimo sledeče:

```
>PIP ime.tip=TI:
prva vrstica datoteke
druga vrstica datoteke
tretja vrstica datoteke
    .
    .
<CTRL/Z>
>
```

S tem smo tvorili novo datoteko na našem direktoriju. Vnos končamo z <CTRL/Z>.

Mnogo boljši je način z editorjem. Editor omogoča poleg pisanja tudi popravljanje. Editorji so posebni pomočni programi za pisanje tekstovnih (ASCII) datotek: programov in drugih tekstov splošnega pomena.

3.3 UREJEVALNIK TEKSTOV EDT(ED2).

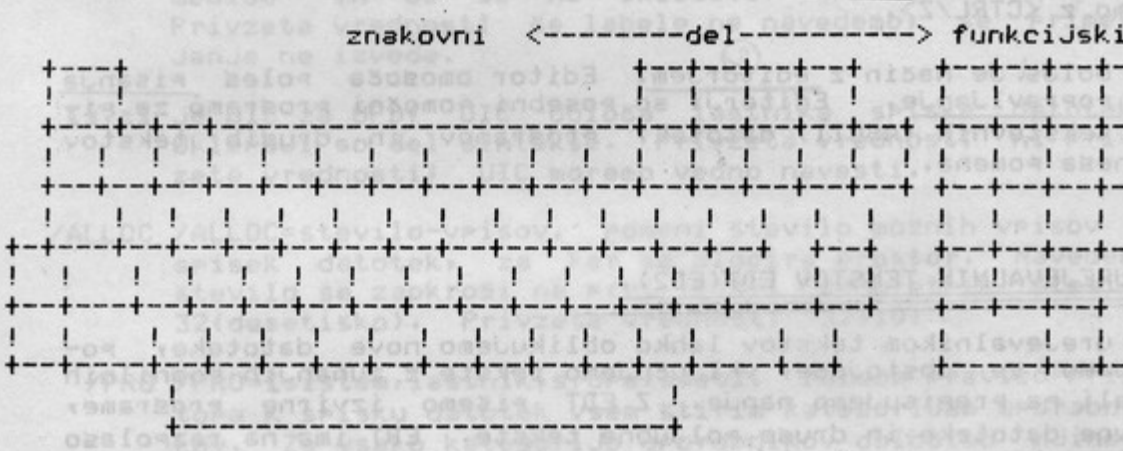
Z EDT urejevalnikom tekstov lahko oblikujemo nove datoteke, popravljamo že obstoječe, vključujemo tekste z zunanjih pomnilnih enot ali pa prepisujemo nanje. Z EDT pišemo izvirne programe, ukazovne datoteke in druge poljubne tekste. EDT ima na razpolago sredstva, ki nam zagotavljajo učinkovitost in enostavno delo. To so:

* HELP - v vsakem trenutku so na voljo kratka pojasnila o funkcijskih tipkah in ukazih
* zaščita našega dela: vse naše operacije s tekstom se vpisujejo v posebno datoteke in v primeru nasilne prekinitve nam ta datoteka služi za restavriranje našega dela.
* dva načina dela: znakovni in vrstični
* številčenje vrstic v vrstičnem načinu
* delo z večimi pomožnimi datotekami
* možnost definiranja funcijskih tipk in kontrolnih znakov, kar nam omogoča da EDT prirediimo svojim potrebam.

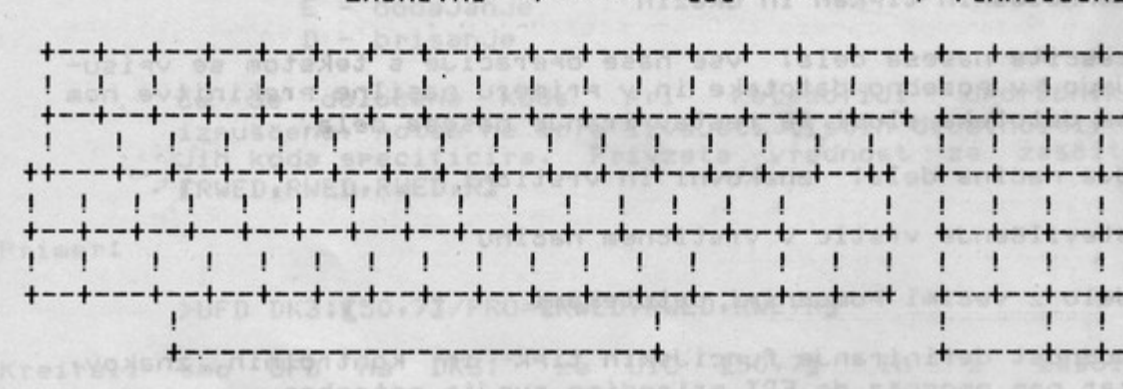
Na tem seminarju se bomo v glavnem omejili na znakovni način dela, ki je možen na terminalih tipa VT100 in VT52. Na terminalih s pomičnim papirjem lahko uporabljamo le vrstični način, zato se bomo seznanili tudi z nekaj ukazi vrstičnega načina.

3.3.1 UPORABA TASTATURE

Tastaturi terminalov tipa VT100 in VT52 se razlikujeta, vendar imata obe dva osnovna dela: znakovni in funkcijski del. Znakovni del tastature je podoben tisti na pisalnih strojih z nekaj dodatnih tipk. Funkcijski del, ki se sicer obnaša kot številska tastatura, ima v urejevalniku EDT poseben pomen. Vsaka tipka opravlja neko funkcijo pri urejanju tekstov. Tastaturi terminalov tipa VT100 in VT52 se razlikujeta predvsem v funkcijskem delu, kjer so nekatere funkcije drugače razvrščene.



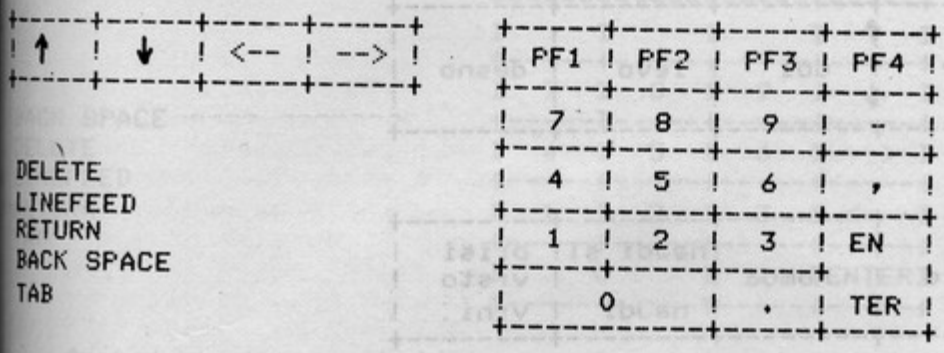
Tastatura terminala tipa VT100



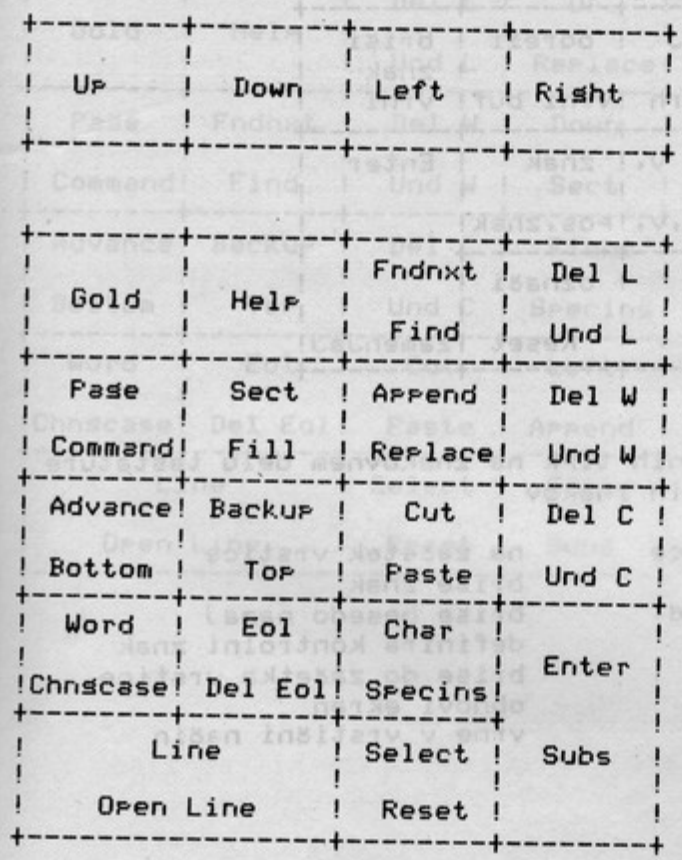
Tastatura terminala tipa VT52

Na znakovnem delu imamo tudi nekaj funkcijskih tipk: to so DELETE, LINE FEED, BACK SPACE, TAB in RETURN. Poleg funkcijskih tipk imajo poseben pomen tudi posamezni kontrolni znaki, ki jih vnašamo tako, da istočasno pritisnemo tipko CTRL in ustrezen znak. V literaturi se kontrolni znaki označujejo z CTRL/znak a na ekranu se nam javlja kot čznak. Kontrolne znake lahko tudi sami definiramo.

Funkcijska tastatura terminala VT100:
izgled tastature



funkcije posameznih tipk v angleščini



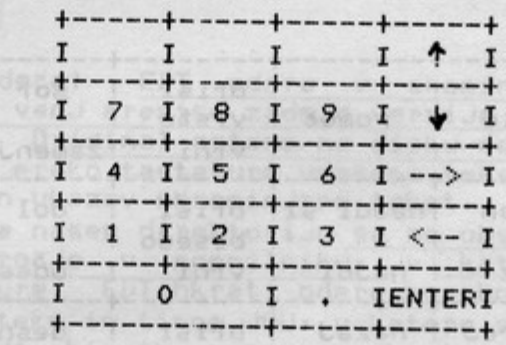
funkcije posameznih tipk v slovenščini

gor	dol	levo	desno
Gold	pomoč	najdi sl / najdi	briši vrsto / vrni
stran ukaz	odsek / polni	dodaj / zamenjaj	briši besedo / vrni
naprej / na dno	nazaj / na vrh	odreži / vrni buf	briši znak / vrni
beseda / pretvori	konec v. / bri.k.v.	znak / pos.znak	Enter
vrsta		označi	
odpri vrsto		Reset	zamenjaj

pomen nekaterih posebnih tipk na znakovnem delu tastature in nekaterih kontrolnih znakov

Backspace	na začetek vrstice
Delete	briše znak
Linefeed	briše besedo nazaj
CTRL/K	definira kontrolni znak
CTRL/U	briše do začetka vrstice
CTRL/W	obnovi ekran
CTRL/Z	vrne v vrstični način

Funkcijska tastatura terminala tipa VT52



BACK SPACE
DELETE
LINEFEED
RETURN
TAB

funkcije posameznih tipk v angleščini

		Del L	Up
Gold	Help		
		Und L	Replace
Page	Fndnxt	Del W	Down
Command	Find	Und W	Sect
Advance	Backup	Del C	Right
Bottom	Top	Und C	Specins
Word	Eol	Cut	Left
Chngcase	Del Eol	Paste	Append
Line		Select	Enter
Open Line		Reset	Subs

funkcije posameznih tipk v slovenščini

Gold	pomoč	briši vrsto vrni	gor zamenjaj
stran ukaz	najdi sl najdi	briši besedo vrni	dol odsek
naprej na dno	nazaj na vrh	briši znak vrni	desno pos.znak
beseda beseda	konec v. konec v.	odreži odreži	levo levo
vrsta		označi	Enter
odpri vrsto		Reset	zamenjaj

pomen posameznih posebnih tipk na znakovnem delu tastature in nekaterih kontrolnih znakov

Backspace	gre na začetek vrste
Delete	briše znak
Linefeed	briše do začetka besede
CTRL/F	polni tekst
CTRL/K	definira kontrolni znak
CTRL/U	briše do zašetka vrste
CTRL/W	obnovi ekran
CTRL/Z	vrne v vrstični način

3.3.2 ZNAKOVNI NAČIN UREJANJA TEKSTOV

Tekst začnemo urejati tako, da izdamo ukaz, ki pokliče EDT. V MCR-u (DELTA/M) ima ukazna vrsta naslednjo obliko:

>EDT enota:[s,č]ime.tip;verzija<CR>

Za imenom urejevalnika napišemo ime datoteke, ki jo želimo urejati. Ime enote lahko izpustimo, če se navedena datoteka nahaja na naši uporabniški sistemski enoti (pseudo ime SY:). [s,č] oz. [seznam] lahko izpustimo, če se datoteka nahaja na našem seznamu in verzijo smemo izpustiti, če želimo popravljati zadnjo verzijo datoteke s tem imenom in tipom. Ime in tip datoteke moramo vedno navesti, saj za to ne obstajata privzeti vrednosti. Če želimo oblikovati novo datoteko na našem seznamu, navedemo ime in tip

datoteke, ki še na njem ne obstajata. Tako navadno pišemo:

```
>EDT ime.tip<CR>
```

V tem primeru se zgodi sledeče: EDT odpre v spominu delovno področje z imenom MAIN in vanj prepiše zadnjo verzijo datoteke z navedenim imenom in tipom. Orisinal ostane na disku nedotaknjen. V delovno področje lahko preko tastature vnašemo tekst ali pa s pomočjo funkcijskih tipk in ukazov spreminjamo tekst. Če datoteka s tem imenom in tipom na našem direktoriju še ne obstaja, ravnotako odpre delovno področje v pomnilniku, v katero potem vnašamo tekst preko tastature. EDT hkrati odpre posebno datoteko journal z imenom naše datoteke in tipom JOU, v katero se vpisujejo vse spremembe in funkcije, ki jih vnašamo preko tastature. Ta journal nam v primeru nasilne prekinitve urejanja služi za restavriranje našega dela.

Poleg glavnega delovnega področja obstaja še delovno področje z imenom PASTE. Sami pa lahko odpiramo še pomožna delovna področja tako, da lahko delamo z večimi vhodnimi in izhodnimi datotekami oziroma nam služijo za manipulacijo z deli datoteke. Vsebino delovnih področij lahko s posebnim ukazom prepišemo na disk in tako oblikujemo datoteko.

Urejanje teksta končamo z ukazom EXIT. Vsebina glavnega delovnega področja se prepiše v datoteko z imenom in tipom, ki smo ga navedli pri klicanju. Številka verzije je najvišja verzija te datoteke plus 1. Ostala delovna področja se brišejo. Če smo oblikovali novo datoteko, dobimo prvo verzijo navedene datoteke. Lahko pa urejanje končamo tako, da delovno področje ne prepišemo na disk. Ko urejanje končamo normalno, se journal sam briše.

Ko začnemo urejati neko tekstovno datoteko z ukazom:

```
>EDT ime.tip
```

se v primeru, ko oblikujemo novo datoteko, izpiše na zaslonu:

```
Input file does not exist

[EOB]

*
```

Če smo navedli ime in tip že obstoječe datoteke, izpiše prvo vrstico te datoteke in v naslednji vrstici zvezdico.

```
1       PRVA VRSTICA DATOTEKE
*
```

EDT se na začetku postavi v vrstični način dela, kar nakazuje zvezdica in utripalka za njo. Vnesemo lahko prvi ukaz. Ker pa bomo delali v znakovnem načinu, vnesemo takoj ukaz za prehod na znakovni način. To je ukaz CHANGE ali kratko (C). Ukaz končamo z <CR>. Zaslon se briše in od vrha navzdol in izpiše se prvih 22

vrstic datoteke (če jih je seveda toliko). Če je datoteka krajša od 22 vrstic (oz. smo oblikovali novo) se za zadnjo vrstico izpiše [EOB].

```
Input file does not exist

[EOB]
*C<CR>

[EOB]

1       PRVA VRSTICA DATOTEKE
*C<CR>

PRVA VRSTICA DATOTEKE
DRUGA VRSTICA DATOTEKE
TRETJA VRSTICA DATOTEKE
CETRTA VRSTICA DATOTEKE
PETA VRSTICA DATOTEKE
SESTA VRSTICA DATOTEKE
SEDMA VRSTICA DATOTEKE
OSMA VRSTICA DATOTEKE
IEOBI
```

in utripalka se pojavi v zgornjem levem kotu zaslona. Sedaj lahko začnemo uporabljati funkcijske tipke ali vnašati tekst.

3.3.2.1 UPORABA FUNKCIJSKE TASTATURE

Tipke funkcijske tastature so označene ali s številkami ali pa z posebnimi znaki. Pri VT100 imamo na vrhu tipke PF1 do PF4, na levi tipke s puščicami, tipke s posebnimi znaki ter ENTER. Pri VT52 imamo poleg številskega dela še puščice, plavo in rdečo tipko, piko ter ENTER. Večina funkcijskih tipk ima dve vrednosti. Zgornja funkcija je dosegljiva z enostavnim pritiskom na ustrezno tipko. Spodnjo dosežemo tako, da najprej pritisnemo funkcijo GOLD (PF1 pri VT100 in plava tipka pri VT52) in šele nato ustrezno tipko.
Primer:

<GOLD> + <BACKUP/TOP>

nam da funkcijo TOP. Če želimo uporabiti funkcijo NAJDI-FIND, moramo pritisniti najprej <GOLD> in šele nato <FINDXT/FIND> tipko.

3.3.2.1.1 POMOČ - funkcija HELP

Funkcija HELP na ekran izpiše diagram funkcijske tastature našega terminala in pomen veljavnih kontrolnih znakov. Če želimo več informacij o posamezni funkcijski tipki, pritisnemo ustrezno

tipko in na zaslonu se izpiše opis te fukcijske tipke. V urejanje se vrnemo tako, da pritisnemo prazen znak. V samih HELP izpisih je tudi navodilo o uporabi HELP funkcije in kako se pride v urejanje.

<HELP>

3.3.2.1.2 Funkcija GOLD

S pomočjo funkcije GOLD so nam dostopne alternativne funkcije funkcijskih tipk. To so tiste, ki so zapisane v drugi vrstici na diagramu, ki ga izpiše HELP funkcija. Alternativne funkcije so nam dosegljive tako, da najprej pritisnemo <GOLD> in nato ustrezno tipko. Primer je naveden zgoraj.

Funkcijo GOLD rabimo tudi, ko želimo določeno funkcijo izvesti ali pa določen znak izpisati večkrat. Najprej pritisnemo GOLD nato vnesemo število in nato funkcijo ali pa znak. Primer:

<GOLD> + <1><0> + <funkcija ali ASCII znak>

in funkcija se izvede 10-krat oziroma se znak izpiše 10-krat. (število 10 vnesemo preko znakovne tastature.)

Tretji primer uporabe: z funkcijo GOLD lahko vnašamo posebne znake (ki jih ni na tastaturi). Najprej pritisnemo <GOLD>, nato ASCII vrednost želenega znaka in funkcijo POSEBNI ZNAK. Primer:

<GOLD> + <2><7> + <GOLD> + <CHAR/SPECINS>

S tem smo vnesli znak ESC, ki ima ASCII vrednost 27.

3.3.2.1.3 Funkcija RESET

Učinek funkcije GOLD (to je, če smo jo pomotoma pritisnili) izbrišemo z funkcijo RESET. Z funkcijo RESET brišemo tudi učinek funkcije OZNAČI-SELECT. Funkcija RESET:

<GOLD> + <SELECT/RESET>

3.3.2.2 VNAŠANJE TEKSTA

Karkoli tipkamo na znakovni tastaturi, ko smo v znakovnem načinu dela, se prenaša v delovno področje na mesto, kjer je utripalka in utripalka se pomakne za eno mesto naprej. Ves tekst za utripalko se ravno tako pomakne v desno. Pravzaprav tekst vrivamo na mestu, kjer je utripalka. Vnos vrstice končamo z <RETURN> oz.

<CR> znakom, ki tako postane zadni znak v vrstici. <CR> na koncu vrstice tudi povzroči, da se odpre za tekočo vrstico nova prazna vrstica. Če <CR> vnesemo nekje v vrstici, se tekst za utripalko skupaj z njo prenese v naslednjo vrstico.

3.3.2.2.1 Funkcija OPEN LINE

S funkcijo OPEN LINE vnesemo znak <CR> za utripalko. Mesto utripalke se ne spremeni. Če je utripalka nekje v sredini vrstice, se tekst za utripalko pomakne na začetek sledeče vrste medtem ko utripalka ne spremeni svoje lege.

<GOLD> + <LINE/OPEN LINE>

3.3.2.3 POMIKANJE UTRIPALKE

Seznanili se bomo, kako lahko pomikamo utripalko po delovnem področju oz. zaslonu.

3.3.2.3.1 Uporaba puščic

Utripalko lahko pomikamo s tipkami, ki so označene s puščicami v smeri, ki jo nakazujejo.

<Gor>

Puščica gor pomakne utripalko na znak v vrstici iznad tekoče. Če je zgornja vrstica krajša, se utripalka postavi na konec zgornje vrstice.

<Dol>

Puščica dol pomakne utripalko na znak izpod trenutnega mesta. Če je spodnja vrstica krajša od trenutne, se postavi na njen konec.

<Levo>

Puščica levo pomakne utripalko za eno mesto v levo v tekoči vrstici. Če se je nahajala na levem robu zaslona, se pomakne na konec zgornje vrstice.

<Desno>

Puščica desno pomakne utripalko za eno mesto v desno. Če se je nahajala za zadnjim znakom v vrstici, se pomakne na začetek sledeče vrste.

3.3.2.3.2 Smer pomikanja

Funkciji NAPREJ-ADVANCE in NAZAJ-BACKUP spremenita smer gibanja utripalke pri uporabi funkcij za pomikanje. Funkcija NAPREJ usmeri gibanja naprej po delovnem področju. Smer naprej velja dokler ne uporabimo funkcije NAZAJ, ki spremeni usmerjenost. Ko začnemo delati v znakovnem načinu, smo usmerjeni naprej.

<ADVANCE/BOTTOM>

uveljavi usmerjenost naprej

<BACKUP/TOP>

uveljavi usmerjenost nazaj

3.3.2.3.3 Pomikanje po tekstovnih enotah

S puščicami se lahko pomikamo v poljubni smeri, vendar je to precej zamudno. Na razpolago imamo posebne funkcije, ki nam omogočajo pomikanje za tekstovno enoto v smeri, v kateri smo usmerjeni.

1. ZNAK-CHAR

pomakne utripalko za eno mesto v veljavni smeri. Če smo na robu se pomaknemo na nasprotni rob naslednje oz. prejšnje vrstice.

<CHAR/SPECINS>

2. BESEDA-WORD

Beseda je skupina znakov, ki ji sledi presledek ali <CR>. Funkcija BESEDA pomakne utripalko na začetek naslednje besede pri usmerjenosti naprej in na začetek tekoče oz. prejšnje besede pri usmerjenosti nazaj. Če se nahajamo na koncu vrste, se pri usmerjenosti naprej pomakne na začetek naslednje vrste. Pri usmerjenosti nazaj se z začetka vrstice pomakne na konec prejšnje.

<WORD/CHNGCASE>

3. KONEC VRSTICE-EOL

pomakne utripalko na konec tekoče vrste pri usmerjenosti naprej in na konec prejšnje pri usmerjenosti nazaj. Pri usmerjenosti naprej in če smo na koncu vrste, se pomaknemo na konec naslednje vrste.

<EOL/DEL EOL>

4. VRSTA-LINE

Pri usmerjenosti naprej se pomaknemo na začetek naslednje vrsta. Pri usmerjenosti nazaj se pomaknemo na začetek tekoče vrste oz. na začetek prejšnje, če smo že na levem robu.

<LINE/OPEN LINE>

5. BACK SPACE

Tipka <BACK SPACE> pomakne utripalko na začetek tekoče vrste oz. prejšnje, če smo že na začetku, neodvisno od usmerjenosti.

<BACK SPACE>

6. SEKCIJA-SECT

Funkcija SEKCIJA pomakne utripalko za 16 vrst v veljavni smeri. Če je v veljavni smeri manj vrstic do konca delovnega področja, javi:

Backup past top of buffer

če je zadel zgorni rob delovnega področja, oz.

Advance past bottom of buffer

če zadene na spodnji rob delovnega področja.

VT100	VT52
<SECT/FILL>	<GOLD> + <DOWN/SECT>

7. STRAN-PAGE

Če imamo v datoteki form-feed (<FF>)znake , ki določajo strani, se po uporabi funkcije STRAN pomaknemo na začetek naslednje strani pri usmerjenosti naprej oz. na začetek tekoče strani pri usmerjenosti nazaj. Če <FF> znakov nimamo v datoteki, se pomaknemo nakonec delovnega področja pri usmerjenosti naprej oz. na začetek pri usmerjenosti nazaj.

<PAGE/COMMAND>

Na konec delovnega področja se lahko pomaknemo tudi z funkcijo NA KONEC-BOTTOM. S funkcijo NA ZAČETEK-TOP se lahko pomaknemo na začetek delovnega področja.

<GOLD> + <ADVANCE/BOTTOM>

pomakne utripalko na konec delovnega področja

<GOLD> + <BACKUP/TOP>

pomakne utripalko na začetek delovnega področja

3.3.2.4 ISKANJE ZAPOREDJA ZNAKOV

S funkcijama NAJDI-FIND in NAJDI NASLEDNJEGA-FNDNXT lahko pomaknemo utripalko na določeno zaporedje znakov. Funkcija FIND nam omogoča definiranje modela, ki ga želimo poiskati in poišče prvo pojavljanje tega modela v želeni smeri.
Pritisnemo:

<GOLD> + <FNDNXT/FIND>

in EDT v zadnji vrstici izpiše:

Search for: model <zaključna funkcija>

Model je poljubno zaporedje znakov (do 64 znakov), ki ga iščemo. Zaključna funkcija pa je lahko NAPREJ-ADVANCE, NAZAJ-BACKUP in ENTER. Če zaključimo model z funkcijo NAPREJ, išče model naprej po delovnem področju, če z NAZAJ išče nazaj po delovnem področju, če pa z ENTER ga išče v trenutno veljavni smeri. Ko model najde, se utripalka ustavi na začetku modela.

Ko pa imamo model že definiran (spravljen v pomnilniku), s funkcijo NAJDI NASLEDNJEGA-FNDNXT poiščemo naslednje pojavljanje tega modela v trenutno veljavni smeri.

<FNDNXT/FIND>

Če modela ne najde do konca oz. začetka delovnega področja, javi na dnu zaslona:

String was not found

Če želimo definirati novi model, moramo spet uporabiti funkcijo FIND.

3.3.2.5 BRISANJE IN VRAČANJE TEKSTA

Skupina funkcij za brisanje nam omogoča brisanje različnih enot teksta. Druga skupina funkcij nam omogoča vračanje pravkar izbrisanih delov. Vsaka funkcija za brisanje ima ustrezno velik pomnilnik, v katerega se zapiše, kar izbrišemo in se tam nahaja

do naslednje uporabe taiste funkcije za brisanje. To nam omogoča, da vsebino tega pomnilnika vrnemo z obratno funkcijo.

3.3.2.5.1 Brisanje in vračanje enega znaka

Za brisanje enega znaka imamo dve funkciji: DELETE in BRIŠI ZNAK-DEL C. DELETE briše znak, ki se nahaja pred utripalko, si ga zapomni in preostanek vrstice pomakne za en znak levo. Tudi utripalka se pomakne za eno mesto v levo. Če smo na začetku vrstice briše <CR> v prejšnji vrsti in primakne tekočo vrsto na konec prejšnje.

<DELETE>

Funkcija BRIŠI ZNAK briše znak na mestu, kjer se nahaja utripalka in znak desno od utripalke pomakne na to mesto. Brisani znak spravi v pomnilnik.

<DEL C/UND C>

Zadnji z DELETE ali BRIŠI ZNAK funkcijo izbrisan znak lahko vrnemo na trenutno mesto utripalke z funkcijo VRNI ZNAK-UND C. Pritisnemo:

<GOLD> + <DEL C/UND C>

in nazadne izbrisani znak se vrne na mesto utripalke.

3.3.2.5.2 Brisanje in vračanje besede

<LINE FEED> briše od utripalke do začetka besede. Če smo na začetku, briše prejšnjo besedo. Izbrisani del besede spravi v pomnilnik za besedo.

<LINE FEED>

Funkcija BRIŠI BESEDO-DEL W briše besedo od utripalke za do konca besede. Če se nahajamo na začetku besede briše celo besedo. Izbrisani tekst spravi v pomnilnik za besedo.

<DEL W/UND W>

Nasprotna funkcija od BRIŠI BESEDO oz. LINE FEED je VRNI BESEDO-UND W, ki vrne na mesto utripalke zadnjo z LINE FEED oz. DEL W izbrisano besedo.

<GOLD> + <DEL W/UND W>

Funkcija je posebej priročna, če je potrebno popraviti le vrstni red besed.

3.3.2.5.3 Brisanje in vračanje vrstice

CTRL/U briše tekst od utripalke pa do začetka vrstice. Izbrisani tekst spravi v vrstični pomnilnik. Če se nahajamo na levem robu, CTRL/U briše prejšnjo vrstico.

Funkcija BRIŠI DO KONCA VRSTE-DEL EOL briše tekst od utripalke pa do <CR> znaka. Če smo na koncu vrste, briše naslednjo vrsto. Brisani tekst spravi v vrstični pomnilnik.

<GOLD> + <EOL/DEL EOL>

Funkcija BRIŠI VRSTO-DEL L briše vrsto od utripalke pa do konca skupaj z <CR> in priključi naslednjo vrsto k tekoči vrst. Izbrisani tekst spravi v vrstični pomnilnik.

<DEL L/UND L>

Funkcija VRNI VRSTICO-UND L vrne na mesto utripalke in na desno tekst, ki se nahaja v vrstičnem pomnilnikom, torej tisto kar smo nazadnje izbrisali z CTRL/U ali DEL EOL ali DEL L funcijo.

<GOLD> + <DEL L/UND L>

3.3.2.6 OZNAČEVANJE IN PREMIKANJE DELOV TEKSTA

Z funkcijami OZNAČI-SELECT, ODREŽI-CUT in VRNI-PASTE lahko dele teksta prenašamo z enega mesta v delovnem področju na drugo ali na več mest v delovnem področju.

Funkcija SELECT nam omogoča, da označimo del teksta ki bi sa želeli izbrisati ali premakniti na drugo mesto. To dosežemo tako, da pritisnemo <SELECT/RESET> fukcijsko tipko, potem pa z funkcijami za premikanje pomikamo utripalko v željeni smeri. Označi se ves tekst od začetnega položaja utripalke pa do končnega. Na terminalih tipa VT100 se označena področja prikažejo obratno, kot črno na belem ozadju, medtem ko se na terminalih tipa VT52 označeni tekst ne razlikuje od običajnega.

<SELECT> + <funkcije za pomikanje ...>

Ko so neko področje označili, lahko uporabimo funcije CUT, APPEND in REPLACE.

Funkcija CUT briše označeni tekst iz glavnega delovnega podočja z imenom MAIN in ga vpiše v delovno področje z imenom PASTE. Dosedanja vsebina delovnega podočja PASTE se briše.

<CUT/PAST>

Če ni označenega področja EDT javi:

No select range active

Namesto funkcije CUT pa lahko uporabimo funkcijo DODAJ-APPEND, ki označeni tekst doda trenutni vsebini PASTE delovnega področja in ga briše iz glavnega delovnega področja.

VT100	VT52
<APPEND/REPLACE>	<GOLD> + <LEFT/APPEND>

Če smo označili neko področje na zaslonu, nam funkcija REPLACE briše ta del iz glavnega delovnega področja in ga nadomesti s trenutno vsebino delovnega področja PASTE.

VT100	VT52
<GOLD> + <APPEND/REPLACE>	<GOLD> + <UP/REPLACE>

Funkcija PASTE vrine od trenutnega mesta utripalke pa naprej vsebino PASTE delovnega področja.

<GOLD> + <CUT/PASTE>

3.3.2.7 ZAMENJEVANJE

Funkciji REPLACE in SUBSTITUTE sta podobni. Eno uporabo tipke REPLACE smo že spoznali. Drug način uporabe te funkcije pa je možen skupaj z FIND oz. FNDNXT. Če smo poiskali nek model (uporabili FIND ali FNDNXT), lahko ta model zamenjamo z vsebino PASTE delovnega področja z pomočjo funkcije REPLACE. Naredimo sledeče:

VT100	VT52
<GOLD> + <FNDNXT/FIND>	<GOLD> + <FNDNXT/FIND>
<ADVANCE/BOTTOM> ali <BACKUPU/TOP>	<ADVANCE/BOTTOM> ali <BACKUPU/TOP>
<GOLD> + <APPEND/REPLACE>	<GOLD> + <GOR/REPLACE>

Ko se zamenjava izvrši, utripalka ostane na koncu z REPLACE izpisanega teksta. Funkcija SUBSTITUTE se od REPLACE razlikuje v toliko, da ko se izmenjava izvrši, se avtomatsko izvede še funkcija FNDNXT in EDT poišče naslednje pojavljanje modela v veljavni smeri.

VT100	VT52
<GOLD> + <FNDNXT/FIND>	<GOLD> + <FNDNXT/FIND>
<ADVANCE/BOTTOM> ali <BACKUPU/TOP>	<ADVANCE/BOTTOM> ali <BACKUPU/TOP>
<GOLD> + <ENTER/SUBS>	<GOLD> + <ENTER/SUBS>

Funkcija SUBSTITUTE je ugodna, če želimo isti model zamenjati večkrat oz. po celem tekstu.

3.3.2.7.1 Zamenjaj male in velike črke

Funkcija ZAMENJAJ ČRKE-CHNGCASE zamenja na označenem področju male in velike črke: male spremeni v velike in obratno.

<SELECT/RESET> + <funkcije pomikanja> +

+ <GOLD> + <WORD/CHNGCASE>

3.3.2.8 FUNKCIJA UKAZ

Funkcija ukaz omogoča vnašanje ukazov. Nas bosta zanimala le dva ukaza: EXIT in QUIT. Funkcija UKAZ-COMMAND

<GOLD> + <PAGE/COMMAND>

povzroči da se na dnu ekrana izpiše odzivnik:

Command:

Vtipkamo ustrezen ukaz in vnos obvezno končamo z ENTER funkcijo. Ukaz EXIT prepiše glavno delovno področje na disk kot datoteko, ki ji da ime in tip vhodne datoteke, verzijo pa poveča za 1. Hkrati se brišejo vsa ostala delovna področja in journal datoteka.

<GOLD> + <PAGE/COMMAND>

Command:EXIT<ENTER/SUBST>

Enako postopamo, ko izdajamo ukaz QUIT, ki pa delovnega področja ne prepiše na disk in torej ne dobimo nove verzije datoteke, ki smo jo urejali. Ostala delovna področja in journal datoteka se brišejo.

Ukaze pa lahko vnašamo tudi v vrstičnem načinu. Iz znakovnega načina pridemo v vrstični z CTRL/Z. Pojavi se zvezdica in lahko izdamo katerikoli ukaz vrstičnega načina dela, tudi EXIT in QUIT.

3.3.3 VRSTIČNI NAČIN DELA Z UREJEVALNIKOM EDT

Kot smo že spoznali, se EDT postavi v vrstični način, ko začnemo delati z njim. Ko smo v znakovnem načinu pa lahko preidemo v vrstični način z CTRL/Z.

V vrstičnem načinu so vrstice oštevilčene od 1 naprej. Pri delu se sklicujemo na te številke oziroma jih uporabljamo za označevanje delov teksta na katere se nanašajo ukazi, ki jih vnašamo. Če vsbino datoteke izpišemo na zaslon, so številke vrstic v prvi koloni kot vidimo na primeru.
Primer:

```
1       PRVA VRSTICA DATOTEKE
2       DRUGA VRSTICA DATOTEKE
3       TRETJA VRSTICA DATOTEKE
4       CETRTA VRSTICA DATOTEKE
5       PETA VRSTICA DATOTEKE
6       SESTA VRSTICA DATOTEKE
7       SEDMA VRSTICA DATOTEKE
8       OSMA VRSTICA DATOTEKE
```

Če za neko vrstico vnesemo nove vrstice, jim EDT priredi vmesne številke. Za vrstico številka 3 bomo vnesli 2 vrstici.

```
1       PRVA VRSTICA DATOTEKE
2       DRUGA VRSTICA DATOTEKE
3       TRETJA VRSTICA DATOTEKE
3.1     PRVA VRINJENA VRSTICA
3.2     DRUGA VRINJENA VRSTICA
4       CETRTA VRSTICA DATOTEKE
5       PETA VRSTICA DATOTEKE
6       SESTA VRSTICA DATOTEKE
7       SEDMA VRSTICA DATOTEKE
8       OSMA VRSTICA DATOTEKE
```

Če smo že večkrat vnašali in brisali, postane oštevilčenje nejasno. Z ukazom RESEQUENS lahko ponovno oštevilčimo vrstice tako, da vmesnih številk ni več. Včasih se ukazi nanašajo na eno vrstico včasih pa na več. Področje imenujemo to, na kar se ukaz nanaša. Področje je lahko povezano ali pa nepovezano. Sestavljeno je iz enega ali več povezanih podočij. Povezano področje označimo tako, da navedemo začetno in končno mejo oz. številko vrstice, če je enovrstično. Za označevanje vrstice uporabljamo:

oznaka	opis
pika (.)	tekoča vrtsica
številka	vrstica s to številko
'niz'	sledeča vrtica, ki vsebuje ta niz
+/- n	vrstica n vrtic naprej/nazaj od tekoče
	privzeta vrednost je +1
m +/- n	vrstica n vrstic naprej/nazaj od
	vrstice m
BEGIN	prva vrstica delovnega področja
END	vrstica za zadnjo vrstico delovnega
	področja
n1:n2	vrstice s številkami n1 do n2, privzeta
	vrednost za n1 oz. n2 je tekoča vrstica
BEFORE	od začetka do tekoče vrstice
REST	od tekoče (vključno) pa do zadnej vrstice
WHOLE	celo delovno področje

Področje je lahko nepovezano, sestavljeno iz večih vrstic. Specificiramo ga tako, da navedemo številke vrstic, ki jih ločimo z vejico: n1,n2,n3,..

3.3.3.1 NEKAJ UKAZOV VRSTIČNEGA NAČINA

Zvezdica (*) v prvi koloni je odzivnik, ki označuje da lahko vnesemo ukaz urejevalniku. Ukaz se navadno nanaša na neko vrstico ali področje. Del ukaza, ki ni obvezen, bomo navajali v oglatem oklepaju. Včasih imajo ukazi kretnice, ki se začnejo z poševno črto. Primer: /QUERY. Nekatere ukaze in kretnice lahko krajšamo. Obvezni del ukazov in kretnic bomo tukaj pisali z velkimi črkami neobvezni pa z malimi. Primer:

Delete [področje][/Query]

3.3.3.1.1 CHANGE

Z ukazom CHANGE preidemo iz vrstičnega na znakovni način dela. Format ukaza:

Change [n]

Neobvezno število n nam omogoča, da utripalko postavimo v n-to vrstico. Če n ne navedemo se utripalka postavi v tekočo vrstico. Obraten ukaz je CTRL/Z.

3.3.3.1.2 TYPE

Ukaz TYPE izpiše navedeno področje na naš ekran. Tekoča vrstica postane prva izpisana vrstica. Format ukaza:

Type [področje],...

Področje lahko specificiramo samo z številko (ena vrstica), n1:n2, n1:END, BEGIN:n2, REST, BEFORE, WHOLE. Privzeta vrednost za številko vrstice, ki smo jo izpustili je tekoča vrstica. Primeri:

```
* T 1:4,7
1       PRVA VRSTICA DATOTEKE
2       DRUGA VRSTICA DATOTEKE
3       TRETJA VRSTICA DATOTEKE
4       CETRTA VRSTICA DATOTEKE
7       SEDMA VRSTICA DATOTEKE
* T 5:6
5       PETA VRSTICA DATOTEKE
6       SESTA VRSTICA DATOTEKE
* T REST
5       PETA VRSTICA DATOTEKE
6       SESTA VRSTICA DATOTEKE
7       SEDMA VRSTICA DATOTEKE
8       OSMA VRSTICA DATOTEKE
* T WHOLE
1       PRVA VRSTICA DATOTEKE
2       DRUGA VRSTICA DATOTEKE
3       TRETJA VRSTICA DATOTEKE
4       CETRTA VRSTICA DATOTEKE
5       PETA VRSTICA DATOTEKE
6       SESTA VRSTICA DATOTEKE
7       SEDMA VRSTICA DATOTEKE
8       OSMA VRSTICA DATOTEKE
```

TYPE ukaza lahko tudi izpustimo. Obvezen je le z WHOLE, REST in BEFORE.

3.3.3.1.3 INSERT

Ukaz INSERT nam omogoča vnašanje teksta v delovno področje. EDT oštevilči vrstice, ki jih insertiramo. Format ukaza:

Insert [n][;vrstica teksta]

Če n navedemo, se tekst vnese pred vrstico številka n. Privzeta vrednost za n je tekoča vrstica. Če navedemo podpičje in vrstico teksta, se vnese samo ta vrstica. Če tega ne navedemo, preidemo v tako imenovani vhodni način in vse kar tipkamo postane del datoteke. V ukazni način se vrnemo z CTRL/Z. Primer:

```
* I 5<CR>
vrinjena vrsta 1
vrinjena vrsta 2
CTRL/Z
* T WHOLE
1         PRVA VRSTICA DATOTEKE
2         DRUGA VRSTICA DATOTEKE
3         TRETJA VRSTICA DATOTEKE
4         CETRTA VRSTICA DATOTEKE
4.1       vrinjena vrsta 1
4.2       vrinjena vrsta 2
5         PETA VRSTICA DATOTEKE
6         SESTA VRSTICA DATOTEKE
7         SEDMA VRSTICA DATOTEKE
8         OSMA VRSTICA DATOTEKE
```

3.3.3.1.4 COPY

Ukaz COPY kopira tekst znotraj delovnega področja. Izhodno področje ostane nespremenjeno. Format ukaza:

```
COpy [področje] TO [n][/Query][/Duplicate:m]
```

Področje, ki ga specificiramo prekopira pred vrstico n. Privzeta vrenost za n je tekoča vrsta. Kretnica QUERY povzroči, da nas EDT za vsako vrstivo iz področja posebej vpraša, ali naj jo prekopira. Kretnica DUPLICATE nam omogoča, da se področje prekopira m-krat. Če duplikate izpustimo, se prekopira enkrat. Primer:

```
* CO 1:3 TO 7
* T WHOLE
1         PRVA VRSTICA DATOTEKE
2         DRUGA VRSTICA DATOTEKE
3         TRETJA VRSTICA DATOTEKE
4         CETRTA VRSTICA DATOTEKE
5         PETA VRSTICA DATOTEKE
6         SESTA VRSTICA DATOTEKE
6.1       PRVA VRSTICA DATOTEKE
6.2       DRUGA VRSTICA DATOTEKE
6.3       TRETJA VRSTICA DATOTEKE
7         SEDMA VRSTICA DATOTEKE
8         OSMA VRSTICA DATOTEKE
```

3.3.3.1.5 MOVE

MOVE ukaz prenese navedeno področje pred navedeno vrsto. Izvirno področje briše. Format ukaza:

Move [področje] TO [n][/Query]

Področje prenese pred vrsto s številko n. QUERY kretnica nam omogoči, da prenesemo samo nekatere vrste iz področja. Privzeta vrednost za področje kakor tudi za n je tekoča vrstica. Primer:

```
* M 4:5 TO 8
* T WHOLE
1         PRVA VRSTICA DATOTEKE
2         DRUGA VRSTICA DATOTEKE
3         TRETJA VRSTICA DATOTEKE
6         SESTA VRSTICA DATOTEKE
7         SEDMA VRSTICA DATOTEKE
7.1       CETRTA VRSTICA DATOTEKE
7.2       PETA VRSTICA DATOTEKE
8         OSMA VRSTICA DATOTEKE
```

3.3.3.1.6 SUBSTITUTE

Ukaz zamenja prvi navedeni niz z drugonavedenim. Niz lahko ima do 64 znakov. Format ukaza:

Substitute/niz1/niz2/[področje][/Query]

Namesto poševne črte smemo uporabiti kak drug posebni znak npr. $. Če področja ne navedemo, se prvo pojavljeni niz1 v tekoči vrstici zamenja z niz2. Če področje navedemo, se na celem specificiranem področju zamenja niz1 z niz2. Kretnica QUERY nam omogoča, da interaktivno določamo ali zamenjavo želimo ali ne. Primer:

```
*S/VRSTICA/vrstica/2:5
2         DRUGA vrstica DATOTEKE
3         TRETJA vrstica DATOTEKE
4         CETRTA vrstica DATOTEKE
5         PETA vrstica DATOTEKE
4 substitution made
*T WHOLE
1         PRVA VRSTICA DATOTEKE
2         DRUGA vrstica DATOTEKE
3         TRETJA vrstica DATOTEKE
4         CETRTA vrstica DATOTEKE
5         PETA vrstica DATOTEKE
6         SESTA VRSTICA DATOTEKE
7         SEDMA VRSTICA DATOTEKE
8         OSMA VRSTICA DATOTEKE
```

3.3.3.1.7 SUBSTITUTE NEXT

Ukaz SUBSTITUTE NEXT zamenja prvo pojavljeni niz1 za niz2 ne glede na to ali se nahaja v tekoči vrsti ali ne. Format ukaza:

```
[Substitute] Next[/niz1/niz2/]
```

Če niz1 in niz2 izpustimo, EDT uporabi zadnji v SUBSTITUTE ukazu naveden niz1 in niz2. Primer:

```
*N/CETRTA/cetrta/
4       cetrta VRSTICA DATOTEKE
*T WHOLE
1       PRVA VRSTICA DATOTEKE
2       DRUGA VRSTICA DATOTEKE
3       TRETJA VRSTICA DATOTEKE
4       cetrta VRSTICA DATOTEKE
5       PETA VRSTICA DATOTEKE
6       SESTA VRSTICA DATOTEKE
7       SEDMA VRSTICA DATOTEKE
8       OSMA VRSTICA DATOTEKE
```

3.3.3.1.8 INCLUDE

Ukaz INCLUDE prepiše datoteko z diska v delovno področje ispred navedene vrstice. Format ukaza:

```
INClude ime-datoteke[področje]
```

Navedena datoteka se prenese v delovno področje ispred prve vrstice navedenega področja ali pred tekočo vrstico.

3.3.3.1.9 WRITE

Ukaz WRITE prepiše navedeno področje na disk in tvori datoteko. Vsebina delovnega področja se ne spremeni.

```
WRite ime-datoteke[področje]
```

Če področja ne navedemo, se prepiše celo delovno področje na disk pod imenom, ki ga navedemo. Primer:

```
*WR TEXT.TMP 2:4
```

3.3.3.10 EXIT

Ukaz EXIT prepiše vsebino delovnega področja z imenom MAIN na disk pod imenom in tipom datoteke, ki smo sa navedli pri startanju urejevalnika. Če pa navedemo ime in tip datoteke, dobi izhodna datoteka tako ime in tip, kot smo sa navedli. Format ukaza:

EXit [ime.tip]

Ko se delovno področje prepiše, imamo na direktoriju novo datoteko in vrnemo se v MCR ali DCL.

3.3.3.11 QUIT

Ukaz QUIT konča urejanje, vendar se delovno področje ne prepiše na disk kot datoteka. Format ukaza:

QUIT[/SAve]

Kretnica SAVE povzroči, da se journal datoteka shrani.

3.3.4 RESTAVRIRANJE

Če je prišlo do nasilne prekinitve urejanja neke datoteke, se v journal datoteki nahajajo vse spremembe, ki smo jih vnesli. Naše delo lahko restavriramo z ukazom:

>EDT ime.tip/RE<CR>

EDT vzita datoteko, ki smo jo navedli in odpre datoteko ime.JOU. Iz ime.JOU bere spremembe ter jih izvaja. Izvajanje vidimo tudi na ekranu. Ko konča, EDT briše datoteko ime.jou ter čaka na naše ukaze.

3.4 PERIFERNI IZMENJEVALNI PROGRAM (PIP)

Periferni izmenjevalni program (PIP) je uporabniški program DELTA/M za prenos podatkovnih struktur z ene standardne enote Files-11 na drugo. Poleg tega izvajamo s PIP-om kontrolne funkcije na datotekah. Glavne funkcije, ki jih izvajamo s PIP-om, so:

- Izpis direktorijev datotek
- Kopiranje datotek
- Izpisovanje datotek na terminal ali tiskalnik
- Brisanje datotek

- Preimenovanje datotek
- Menjanje zaščite datotek
- Deblokiranje datotek

Uporabniški program PIP kličemo na dva načina:

1. >PIP ukazni-niz

   Izza privzetega ali pa eksplicitnega MCR> odzivnega znaka napišemo PIP, presledek in ukazni-niz.

2. >PIP<CR>
   PIP>ukazni-niz

   Pokličemo PIP-ov odzivni znak in nato dodamo ukazni-niz. PIP mora biti instaliran.

S kretnicami določamo, katera operacija se bo izvedla.

3.4.1 UKAZNI NIZ PIP

Funkcije PIP zahtevamo z uporabo ukaznih nizov PIP-a preko terminala oziroma datoteke za indirektne ukaze. Oblike ukaznih nizov so odvisne od funkcije, katero izvajamo, in so zato opisane v posebnih podpoglavjih.

3.4.1.1 PRIVZETE VREDNOSTI PRI SPECIFICIRANJU IMENA DATOTEKE

PIP privzame zadnjo vneseno vrednost v ukaznem nizu. To pomeni, da PIP vnešene vrednosti privzame in jih spremeni, ko jih vi spremenite. Izjeme k temu pravilu so opisane v opisih posameznih kretnic.

Z naslednjim ukaznim nizom postavimo novo privzeto vrednost za vsako datoteko iz ukaznega niza.

```
>PIP T1.MAC;5, T2,.TSK,;6/BR
T1.MAC;5
T2.MAC;5
T2.TSK;5
T2.TSK;6
```

V tabeli so opisana pravila, katera uporablja PIP za postavljanje privzetih vrednosti.

PIP privzete vrednosti za specifikacije datotek

Element	Privzeta vrednost
dev:	Uporabnikova sistemska enota SY:, ki mu jo je dodelil sistem ob vključitvi oz. ki si jo je uporabnik priredil s ukazom ASN ali specificiral s PIP-ovo kretnico /DF. Za specifikacijo več datotek PIP upošteva predhodno privzeto enoto, če ni nova enota eksplicitno specificirana.
[ufd]	Je UIC, pod katerim ste se prijavili, oz. UIC definiran z /DF. Pri specifikaciji več datotek PIP upošteva predhodno navedeni ali privzeti ufd, če ni nov ufd eksplicitno specificiran. Dovoljena je uporaba zvezdic.
ime	Ni privzete vrednosti za specifikacijo prve datoteke. Za nadaljnje datoteke je privzeta vrednost zadnje eksplicitno specificirane datoteke. Dovoljena je uporaba specifikacije z zvezdicami.
tip	Ni privzete vrednosti za specifikacijo prve datoteke. Za specifikacijo ostalih datotek je privzeta vrednost zadnjega tipa eksplicitno specificirane datoteke. Dovoljena je uporaba specifikacije z zvezdicami.
verzija	Za vhodne datoteke je privzeta vrednost zadnje verzije. Za izhodne datoteke je privzeta vrednost naslednje število verzije, oziroma verzija 1, če datoteka še ne obstaja na direktoriju. Izjema je funkcija PIP za brisanje datotek, ki zahteva ali eksplicitno navedbo verzije ali pa uporabo zvezdice. Številka verzije mora biti v obliki ;n, kjer je n večje od 0 (n>0). Za specifikacijo najstarejše verzije je možna notacija ;-1. Verzija ;0 ali ; se lahko uporabi za označevanje zadnje verzije. V določenih primerih se lahko uporabi zvezdica

3.4.1.2 KRETNICE PIP-a

Delovanje PIP-a kontroliramo s kretnicami in podkretnicami. Kretnica se sestoji iz poševne črte (/) in same kretnice, je navadni dvoznakovna. Nekatere kretnice imajo lahko tudi številski

ali pa znakovni argument, ki sa vnašamo za dvopičjem: /kretnica:argument. Podkretnice lahko detajlni opredelijo delovanje kretnice. Navajamo jih za kretnico, ki jo spremljajo.

Vse kretnice v PIP-u delujejo na zaporedje specificiranih datotek. Nekatere kretnice lahko navedemo tudi brez navedenih datotek. Kretnica se ne sme nahajati ispred imena datoteke, na katere naj deluje. Lahko pa deluje na več datotek hkrati. Primer: Kretnica /PU deluje na vse tri specificirane datoteke.

```
PIP dat1,dat2,dat3/PU
```

Podkretnice delujejo samo lokalno na datoteko, ki ji sledijo. Argumenti kretnic so lahko zapisani oktalno (privzeta vrednost), ali decimalno (, če jih navedemo s piko.

Kretnice in podkretnice PIP-a

Kretnica	Podkretnica	Funkcija
/AP		Doda datoteko na ze obstoječo datoteko
	/FO	Določa lastnika datoteke
/BR		Izpis direktorija v zgoščeni obliki
	/CD	Pri kopiranju datum kreiranja vhodne datoteke postane tudi datum kreiranja izhodne datoteke in ne kot sicer trenutni datum.
/DE		Brisanje ene ali več datotek
	/LD	Izpis brisanih datotek
/DF		Spreminjanje privzete enote in/ali UFD
/FR		Izpis, koliko nezasedenega prostora je na določeni enoti in dolžino najdaljšega zaporednega prostora na tej enoti
/FU:n		Izpis direktorija v popolnem formatu (alternativa kretnice /LI).
/LI		Izpis direktorija.
/ME		Združevanje dveh ali več datotek v eno.
	/CO	Določa, da bo izhodna datoteka zapisana v zaporednih blokih.
	/FO	Specificira lastnika datoteke.
	/NV	Poveča se številka zadnje verzije za 1.

	/SU	Zamenja se obstoječa datoteka.
/PR		Sprememba zaščite datoteke.
	/FO	Specificira lastnika datoteke.
	/GR:RWED	Postavi se zaščita za čitanje/pisanje/razširjanje/brisanje na nivoju skupine.
	/OW:RWED	Postavi se zaščita za čitanje/pisanje/razširjanje/brisanje na nivoju lastnika
	/SY:RWED	Postavi se zaščita za čitanje/pisanje/razširjanje/brisanje na nivoju sistema
	/WO:RWED	Postavi se zaščita za čitanje/pisanje/razširjanje/brisanje na nivoju ostalih (world)
/PU:n		Brisanje starih verzij datotek.
	/LD	Izpis brisanih datotek.
/RE		Preimenovanje datoteke.
/RW		Previjanje magnetnega traku.
/SD		Selektivno brisanje datotek.
/SP		Pisanje datoteke na vrstičnem tiskalniku.
/TB		Izpis skupnega števila uporabljenih blokov v direktoriju števila alociranih blokov in število datotek v direktoriju.
/UN		Deblokira datoteke.

3.4.1.3 ZVEZDICE

PIP dovoljuje uporabo znaka zvezdica (*) v specifikaciji datoteke. Znak zvezdica (*), na enem ali več mestih specifikacije datoteke, pomeni "vse", kot na primer vse tipe datoteke. Uporaba zvezdic ni poljubna. V naslednjih podpoglavjih bo opisana uporaba zvezdic za vhodne in izhodne datoteke.

3.4.1.3.1 Zvezdice v specifikaciji izhodne datoteke

Uporaba zvezdic v specifikaciji elementov izhodne datoteke je omejena. V naslednjih funkcijah PIP ni dovoljena uporaba zvezdic v specifikaciji izhodne datoteke:

- kopiranje posamezne datoteke
- združevanje datotek v novo datoteko
- dodajanje k obstoječi datoteki
- ažuriranje obstoječe datoteke
- izpis direktorija

Uporaba zvezdic v specifikaciji izhodne datoteke za katerokoli izmed zgoraj navedenih funkcij lahko povzroči, da je ukazna vrsta nerazumljiva, oziroma da vsebuje neskončno število izhodnih datotek. Na primer:

```
PIP [200,200]*.*;*=TEST.DAT
```

PIP bo poizkusil tvoriti neskončno število datotek na [200,200] iz ene same datoteke. Kadar želite kopirati več datotek, mora biti izhodna specifikacija v obliki *.*;* ali privzete vrednosti.

Za kretnico preimenuj (/RE) in vnesi (/EN) lahko v izhodni specifikaciji vnašamo zvezdice v kombinaciji s specificiranimi polji.

3.4.1.3.2 Zvezdice v specifikaciji vhodne datoteke

PIP dovoljuje naslednje oblike uporabe zvezdic za specifikacijo vhodnih datotek:

- *.*;* pomeni vse verzije vseh datotek
- *.MAC;* pomeni vse verzije vseh datotek tipa MAC
- PROG.*;* pomeni vse verzije vseh tipov datoteke PROG
- PROG.MAC;* pomeni vse verzije datoteke PROG.MAC
- *.* pomeni zadnje verzije vseh datotek
- *.FTN pomeni zadnje verzije vseh datotek tipa FTN
- PROG.* pomeni zadnje verzije vseh tipov datoteke PROG

PIP omogoča tudi naslednje oblike uporabe zvezdic za UFD:

- [*,*] pomeni vse direktorije
- [m,*] pomeni vse člane skupine m
- [*,k] pomeni vse skupine s članom k

3.4.2 LISTANJE DIREKTORIJEV

Za listanje direktorijev imamo več kretnic, ki nam dajejo različen obseg informacij o datotekah na direktoriju. Najbolj pogosto uporabljamo kretnico /LI.

S kretnico /LI izpišemo ime eno ali več datotek iz uporabnikovega direktorija, skupaj z ostalimi informacijami o datoteki. PIP pozna še tri kretnice za izpis direktorijev (/BR, /FU in /TB). Kretnico /LI uporabljamo na naslednji način:

[listdat]=dat1[,dat2,...,datn]/LI

kjer so:

listdat Datoteka v kateri bo izpis, specificirana v obliki:

enota:[g,č]ime.tip;verzija

če izpisa datoteke ne navedemo, je privzeta vrednost TI:

datN To je ime vhodne datoteke v naslednji obliki:

enota:[g,č]ime.tip;verzija

Privzeta vrednost za vhodno datoteko je *.*;*.

/LI Kretnica s pomočjo katere dobimo naslednje informacije:

```
      PIP [150,3]/LI
(1)   DIRECTORY DE1:[150,3]
      21-APR-83 11:30
      (2)                (3)     (4)     (5)
      GGPLA.OBJ;1         8.             21-APR-83 08:02
      GGPLA.SKL;1         1.             21-APR-83 08:02
      SREDA.;1            7.             21-APR-83 08:02
      GGPLA.CBL;13        4.             21-APR-83 08:02
      GGPLA.CBL;12        4.             19-APR-83 14:41
      GGPLA.TSK;2         103.    C      19-APR-83 14:51
      TOREK.;2            7.             19-APR-83 14:51

(6)   TOTAL OF 134./135. BLOCKS IN 7. FILES
```

(1) enota, UFD, datum in čas izpisa
(2) ime.tip;verzija datoteke
(3) število zavzetih blokov (decimalno)
(4) koda datoteke: datoteka je
(nula)=ni povezana
C =povezana (contiguous)
L =nepravilno zaprta
(5) datum in čas kreiranja
(6) skupno število zavzetih/alociranih blokov za izlistane datoteke.

namesto kretnice /LI lahko uporabimo /BR, /FU, /TB ali /FR.

Z /BR kretnico dobimo manj informacija; samo ime, tip in verzijo in kateri direktorij smo izpisali.

```
      PIP [150,5]/BR
```

```
DIRECTORY DE1:[150,3]

GGPLA.OBJ;1
GGPLA.SKL;1
SREDA.;1
GGPLA.CBL;13
GGPLA.CBL;12
GGPLA.TSK;2
TOREK.;2
```

Z /FU kretnico dobimo detaljni opis datoteka, katerih imena izlistamo. Za datoteke, katerih imena želimo izlistati, moramo imati pravico čitanja.

```
PIP [150,3]/FU
DIRECTORY DE1:[150,3]    (1)
21-APR-83 11:30
   (2)                (3)          (4)   (5)      (6)
GGPLA.OBJ;1         (1442,51)      8./8.        21-APR-83 08:02
 [150,3]  [RWED,RWED,RWED,R] (7)
GGPLA.SKL;1         (1453,102)     1./1.        21-APR-83 08:02
 [150,3]  [RWED,RWED,RWED,R]
SREDA.;1            (1526,51)      7./7.        21-APR-83 08:02
 [150,3]  [RWED,RWED,RWED,R]
GGPLA.CBL;13        (1530,41)      4./4.        21-APR-83 08:02
 [150,3]  [RWED,RWED,RWED,R]
GGPLA.CBL;12        (2437,56)      4./5.        19-APR-83 14:41
 [150,3]  [RWED,RWED,RWED,R]
GGPLA.TSK;2         (2776,7)     103./103. C    19-APR-83 14:51
 [150,3]  [RWED,RWED,RWED,R]    19-APR-83 14:51(2.) (8)
TOREK.;2            (3046,54)      7./7.        19-APR-83 14:51
 [150,3]  [RWED,RWED,RWED,R]    19-APR-83 15:02(2.)

TOTAL OF 134./135. BLOCKS IN 7. FILES
(9)
```

(1) enota, UFD, datum in čas izpisa
(2) ime.tip;verzija datoteke
(3) identifikacijsko število: ID in SQ števili
(4) število zavzetih/alociranih blokov (decimalno)
(5) koda datoteke: datoteka je
    (nula)=ni povezana
    C =povezana (contiguous)
    L =nepravilno zaprta
(6) datum in čas kreiranja
(7) UIC lastnika datoteke v obliki [s,č] in zaščita datoteke za vse štiri kategorije uporabnikov:
    [privilegirani,lastnik,grupa,ostali]
    Pri vsaki kategoriji je navedeno kaj uporabnik sme delati. Oznaka
    R označuje, da sme brati
    W označuje, d sme pisati
    E označuje, da sme širiti
    D označuje, da sme brisati
    če je neka oznaka pri neki kategoriji izpuščena, to

pomeni da ji je ta dejavnost prepovedana.
(8) datum in čas zadnje spremembe v datoteki in zaporedna številka spremembe.
(9) Skupno število zavzetih/alociranih blokov za izlistane datoteke.

Kretnica /TB izpiše število zavzetih/alociranih blokov za datoteke, ki smo jih navedli.

```
PIP [150,3]/TB

STORAGE USED/ALLOCATED FOR DIRECTORY DR1:[303,205]
17-MAY-84 11:20

TOTAL OF 134./135. BLOCKS IN 7. FILES
```

S kretnico /FR lahko zvemo, koliko imamo na določeni enoti vseh blokov, koliko prostih, in koliko je največji povezani prostor na disku (disketi). Kretnico uporabimo na sledeči način:

```
PIP [enota:]/FR
```

če enote ne navedemo izpiše za SY:. Odgovor ima sledečo obliko:

```
enota: HASxxxx. BLOCKS FREE, yyyy.BLOCKS USED OUT OF ZZZZ.
LARGEST CONTIGUOUS SPACE = nnnn.BLOCKS
```

Primeri:

1. PIP>/LI
   Izpiše se direktorij z našim trenutnim UIC z naše SY: enote. To ustreza ukazu TI:=*.*;*/LI.

2. PIP>LP:=[*,*]/FU
   Na tiskalniku se izpišejo vsi direktoriji razen [0,0] z naše SY: enote v polnem formatu.

3. PIP>TI:=TEST.DAT/FU
   Na TI: se v polnem formatu izpiše zadnja verzija datoteke TEST.DAT z našeg UIC-a in naše SY: enote.

4. PIP APR19.DAT=[200,200]*.*/LI
   Direktorij zadnjih verzij vseh datotek z UIC [200,200] in SY: enote se vpiše v datoteko APR19.DAT na naš UFD na isti enoti.

3.4.3 KOPIRANJE DATOTEK

Datoteke tipa Files-11 lahko kopiramo z ukazi PIP brez uporabe kretnic. Pri kopiranju imamo vedno eno ali več vhodnih datotek in ravno toliko izhodnih datotek. Izhodne datoteke so navedene na levi strani enačaja, vhodne pa na desni. Na desni strani

lahko navedemo več datotek. Ukazna vrstica ima obliko:

>PIP izh-dat[/podkr]=vh-dat1[/podkr],...,vh-datN[/podkr]

izh-dat Pri izhodni datoteki smemo navesti samo eno datoteko v obliki:

enota:[s,č]ime.tip;verzija

Če enoto in/ali UFD izpustimo se upoštevajo privzete vrednosti. Če navedemo namesto imena, tipa in verzije *.*;* ali vse skupaj izpustimo se privzamejo imena, tipi in verzije vhodnih datotek (slej podkretnici NV in SU). V primeru, da vnesemo katerekakoli od elementov izhodne datoteke (ime ali tip ali verzijo), ni možna uporaba zvezdic v taki specifikaciji. To pomeni, da za kopiranje, pri katerem je definiran vsaj en element specifikacije izhodne datoteke, moremo definirati le eno vhodno datoteko. Če pa navedemo več vhodnih datotek in le eno izhodno, se izvrši združevanje datotek po takem vrstnem redu, kot smo jih navedli.

vh-datI ena ali več vhodnih datotek v obliki:

enota:[s,č]ime.tip;verzija

Če izpustimo ime, tip in verzijo, je privzeta vrednost *.*;*. Če izpustimo enoto in/ali UFD, PIP upošteva privzete vrednosti. V specifikacijah lahko poljubno uporabljamo zvezdice.

/podkr Podkretnice lahko navedemo na izhodni strani in potem veljajo za vse prepisane datoteke, ali pa na vhodni strani za specificirano datoteko in potem velja samo zanjo (oz. skupino, če so uporabili zvezdice).

Podkretnice za kopiranje in združevanje

Podkretnica	Opis
/CD	Datum kreiranja vhodne datoteke se prenese na izhodno datoteko. Če sa izpustimo, dobi kopija kot datum kreiranja datum prepisa.
/CO	S to podkkretnico zahtevamo, da naj bo izhodna datoteka zapisana v zaporednih blokih.
/-CO	S to podkretnico dovoljujemo poljuben zapis izhodne datoteke. Če ne navedemo zgornjih podkretnic privzame PIP vrednosti atributov, ki veljajo za vhodno datoteko.
/FO	S to podkretnico specificiramo, da je ~~lastnik~~ lastnik izhodne datoteke lastnik direktorija na katerega pišemo. Če te podkretnice ne navedemo, je lastnik prepisanih datotek UIC, pod katerim prepisovanje izvajamo.
/NV	Nova verzija - ta podkretnica poveča številko verzije izhodne datoteke. Če datoteka, ki jo kopiramo, že obstaja v izhodnem direktoriju, postane se številka verzije poveča za ena večja od najvišje verzije. Če pa datoteke še ni v izhodnem direktoriju, se ji priredi verzija ena.
/SU	S to podkretnico omogočimo kopiranje ene ali več vhodnih datotek v datoteko, katere ime, tip in verzija že obstajata na UFD-ju. Obstoječa datoteka se zbriše, nova se tvori iz podatkov vhodne datoteke. Pri tem se spremeni le identifikacijsko število datoteke. Možne so tudi spremembe v atributih in številu rezerviranih blokov.

Primeri:

1. PIP>DE1:VZOREC.DAT=DE0:TEST.DAT
   S tem ukazom kopiramo zadnjo verzijo datoteke TEST.DAT (na delovnem UFD) z DE0 na DE1: pod imenom VZOREC.DAT.

2. PIP>DE1:[*,*]=DE0:[15,*]
   S tem ukazom prekopiramo vse datoteke vseh članov skupine 15 na DE0: na ustrezne UFD-je na DE1:

3. PIP>DE1:VZOREC.DAT=DE0:TEST.DAT;1,PODAT.DAT;2

S tem ukazom združimo verzijo 1 datoteke TEST.DAT in verzijo 2 datoteke PODAT.DAT z enote DE0: v datoteko VZOREC.DAT na enoto DE1:, vse tri datoteke so na trenutnem UFD-ju.

4. PIP>DK1:=DK0:*.FTN:

S tem ukazom prepišemo na trenutni UFD vse verzije vseh datotek tipa .DAT z DK0: na DK1:

5. PIP>DK0:[200,200]=DK:[200,220]TEST.DAT/FO

Lastnik izhodne datoteke je UIC [200,200]. Ko uporabljamo kretnico /FO moramo imeti pravico pisanja na izhodni UFD.

6. PIP>DK1:[*,*]/NV=DK:[*,10]*.MAC/FO

Zadnje verzije datotek tipa .MAC z vseh direktorijev člana 10 z enote DK: prepišemo na ustrezne direktorije na DK1:. Lastniki izhodnih datotek so lastniki izhodnih direktorijev. Kreirajo se nove verzije.

7.
```
DIRECTORY DE1: [100,62]
26-MAY-80 13:24

PODAT.DAT;21              13.     3-APR-80   12:47

PIP>TEST1.DAT/CD=PODAT.DAT

DIRECTORY DE1:[100,62]
26-MAY-80 13:26

PODAT.DAT;21              13.     3-APR-80   12:47
TEST1.DAT;1               13.     3-APR-80   12:47
```

Iz datoteke PODAT.DAT smo dobili novo datoteko TEST.DAT z datumom, ki pripada datumu datoteke PODAT.DAT in ne datumu prenosa.

3.4.3.1 Dodajanje

Poseben primer kopiranja je tudi dodajanje vsebine ene ali večih datotek že obstoječi datoteki. V ta namen uporabimo kretnico /AP. Kretnica /AP odpre že obstoječo datoteko in doda eno ali več vhodnih datotek na konec ze obstoječe datoteke v takem vrstnem redu kot smo jih navedli oz. kot so navedene v direktoriju, če smo uporabili zvezdice

PIP>izh.dat=vh-dat1,....vh-dat.n/AP[/FO]

Izhodna datoteka mora obstajati. Vse navedene vhodne datoteke se dodajo. Podkretnica /FO ima isti pomen kot pri kopiranju. Uporaba zvezdic ni dovoljena na mestu izhodne datoteke.

Primer:

```
PIP>DE1:DAT1.DAT;1=DAT2.DAT;1,DAT3.DAT;1,DAT4.DAT;1/AP
```

Odpre datoteko DAT na DE1: in ji doda datoteke DAT2, DAT3, in DAT4.

3.4.4 IZPISOVANJE ASCII DATOTEK

Datoteke lahko izpisujemo na printerju ali pa terminalu. Na printerjulahko izpisujemo direktno ali pa s pomočjo urejevalnika izpisne vrste (seveda česa naš sistem ima).

Direktno izpišemo na printer ali terminal izpišemo z ukazom

```
PIP enota:=datoteka1,datoteka2,....
```

Dovoljena je uporaba zvezdic v imenu datoteke. Enote pa je lahko printer (LPn:), terminal TTn: (n je številka enote tega tipa v sistemu) oz. pseudo terminal (TI:). Datoteke se bodo izpisale na navedeni enoti v vrstnem redu, kot smo jih navedli oziroma v takem vrstnem redu kot so zapisane na direktoriju, če smo uporabili zvezdice. Če je aktiven Queue Manager, na ta način ne moremo pošiljati datotek na tiskalnik.

Izpis na printer preko urejevalnika vrste pa dosežemo z kretnico (/SP). Ta kretnica deluje le, če je instaliran program PRINT ali pa je aktiven QUEUE MANAGER. Kretnico uporabimo na naslednji način:

```
>PIP vh-dat[,....,vh-datN]/SP[:n[.]]
```

vh-datI Specifikacija vhodne datoteke, ki jo želimo izpisati v obliki:

```
enota:[s,č]ime.tip;verzija
```

Vhodno datoteko moramo specificirati, ker prazna polja za ime, tip ali verzijo ne privzamejo vrednosti *.*;*.

/SP kretnica /SP

n Pomeni število kopij, kajti jih želimo izpisati. Če n ne navedemo, se privzame vrednost 1.

Primer:

```
PIP>A.LST;1,BB.LST;2/SP
```

Izpišeta se datoteki A.LST in BB.LST;2.

3.4.5 BRISANJE DATOTEK

Za brisanje datotek imamo tri kretnice /DE, /SD in /PU ter nekaj podkretnic.

3.4.5.1 Kretnica /DE

S kretnico /DE brišemo datoteke iz direktorija. S podkretnico /LD lahko zahtevamo, da se imena brisanih datotek izpišejo na terminalu. Za datoteke, ki jih želimo izbrisati, moramo imeti pravico brisanja. Kretnico /DE uporabljamo na naslednji način:

```
>PIP dat1[,dat2,.....datN]/DE[/LD]
```

kjer so:

datI Ime datoteke, ki jo želimo brisati. Enoto in UFD smemo izpustiti in privzame se enota SY: in trenutni UFD. Verzijo moramo obvezno navesti ali pa napisati zvezdico.

/DE Kretnica za brisanje.

/LD podkretnica za izpis imen brisanih datotek.

Če zelimo brisati najstarejšo verzijo, uporabimo število ;-1, če želimo brisati zadnjo verzijo, uporabimo število ;0 ali ;_.

Primeri:

1. PIP>TEST.MAC;-1/DE
   Briše se najstarejša verzija datoteke TEST.MAC.

2. PIP>TEST1.DAT;0, TEST2.DAT;/DE
   Brišeta se zadnji verziji datotek TEST1.DAT in TEST2.DAT

Kadar uporabljamo verzijo ;-1,;0 in ; ni dovoljena uporaba * za specifikacijo imena ali tipa datoteke.

Če je verzija specificirana eksplicitno ali z *, je možna poljubna oblika specifikacije vhodne datoteke, vključujoč tudi zvezdice in to celo v specifikacijo skupine in člana skupine.

Primeri:

1. PIP>TEST1.DAT;1,;2/DE
   Brišeta se verziji 1 in 2 datoteke TEST1.DAT na izbrani enoti in direktoriju.

2. PIP>TEST.DAT;5/DE
   Briše se verzija 5 datoteke TEST.DAT na izbrani enoti in direktoriju.

3. PIP>*.OBJ;*,*.TMP;*/DE/LD
   Brišejo se vse verzije vseh datotek tipa .OBJ in .TMP z izbrane enote in direktorija. Imena vseh brisanih

datotek tipa .TMP se izpišejo na terminalu.

3.4.5.2 Selektivno brisanje /SD

Pri selektivnem brisanju navadno navedemo skupino datotek (s pomočjo zvezdic) od katerih želimo nekatere brisati. PIP nampo vrsti izpisuje imena datotek iz navedene skupine in mi se pri vsaki odločamo ali jo brišemo ali ne. Možni so štirje odgovori in dva terminatorja. V tabeli so pisani vsi možni odgovori in njihovo delovanje. Kretnico /SD uporabljamo na naslednji način:

PIP>dat1[,dat2...datN]/SD

kjer so:

datI To je specifikacija datoteke v obliki:

enota:[s,č]ime.tip;verzija

Datoteko moramo specificirati, ker prazna mesta polja za ime, tip ali verzijo ne privzamejo vrednosti *.*;*.

/SD Kretnica za selektivno brisanje.

Možni odgovori pri selektivnem brisanju

Črka	Terminator operacije	Opis
Y	(<CR>)	briši to datoteko in nadaljuj
Y	(<CTRL/Z)	briši to datoteko in izstopi iz PIP-a
N	(<CR>)	shrani to datoteko in nadaljuj
N	(<CTRL/Z)	shrani to datoteko in izstopi iz PIP-a
	(<CR>)	shrani to datoteko in nadaljuj
	(<CTRL/Z)	shrani to datoteko in izstopi iz PIP-a
Q	(<CR>)	shrani to datoteko in se vrni
Q	(<CTRL/Z)	shrani to datoteko in izstopi iz PIP-a
G	(<CR>)	briši to datoteko in vse ostale kandidate za brisanje, izpiši brisane datoteke in vrni se v PIP komandno obl.
G	(<CTRL/Z)	briši to datoteko in vse ostale kandidate za brisanje, izpiši brisane datoteke in izstopi iz PIP-a

Primeri:

1. PIP PROGR.DAT;*/SD
   DELETE FILE DE0:[110,110]PROGR.DAT;1 Y/N/Q/G? Y(<CR>)
   DELETE FILE DE0:[110,110]PROGR.DAT;2 Y/N/Q/G? G(<CR>)
   THE FOLLOWING FILES HAVE BEEN DELETED:
   DE0:[110,110]PROGR.DAT;2

DEO:[110,110]PROGR.DAT;3

Briše se datoteka PROGR.DAT;1 in PIP gre do naslednjega kandidata - datoteke PROGR.DAT;2. Briše se ta in vse ostale verzije datoteke PROGR.DAT. Izpišejo se brisane datoteke in vrnemo se v PIP komandno obliko.

2. PIP TEST.*;*/SD
DELETE FILE DEO: [100,100]TEST.MAC;1 Y/N/G/Q ? N(<CR>)
DELETE FILE DEO: [100,100]TEST.OBJ;1 Y/N/G/Q ? Q(<CTRL/Z)

Ohrani se datoteka TEST.MAC;1 in PIP gre do naslednjega kandidata-datoteke TEST.OBJ;1. Ta in vse ostale datoteke z imenom TEST se ohranijo. Vrnemo se v kontrolno obliko monitorja.

3.4.5.3 Brisanje starih verzij datotek

S /PU kretnico brišemo določeno število starih verzij datotek. Lahko tudi specificiramo, da brisane datoteke izpišemo na terminalu. Kretnico /PU uporabljamo na naslednji način:

PIP>dat1[,dat2,...datN]/PU[:n][/LD]

kjer so:

datI ime vhodne datoteke v obliki:

enota:[g,č]ime.tip

/PU:n kretnica za brisanje. Če navedemo parameter n in je število verzij datoteke m, bodo brisane vse obstoječe datoteke s številko verzije manjšo ali enako m-n . Čeprav pravimo, da s to kretnico brišemo vse razen n zadnjih verzij datoteke, je važno, da se zavedamo, da v primeru, če so že brisane katere verzije med m-n in m, nam bo ostalo manj kot n verzij. Zadnja verzija datoteke se vedno ohrani.

Če n ne navedemo, privzame PIP za n vrednost 1 in se brišejo vse razen zadnje verzije datoteke. Če je n večji od števila verzij datoteke, se ne briše nobena verzija.

Vrednost za n je oktalna. Če navedemo n na koncu ukazne vrste, se ta n nanaša le na zadnjo datoteko v ukazni vrsti. Za vse ostale datoteke se privzame vrednost 1. Seveda se n nanaša na vse naslednje datoteke, dokler ne spremenimo njegove vrednosti.

/LD podkretnica s katero dosežemo, da se izpišejo imena brisanih datotek na terminal.

Primeri:

1. PIP>*.OBJ,*.MAC/PU:2/LD
   Brišejo se vse razen zadnje verzije datotek tipa .OBJ in vse razen zadnih dveh verzij datotek tipa .MAC. Izpišejo se vse brisane verzije datotek tipa .MAC.

2. PIP>*.OBJ/PU:2/LD,*.MAC
   Brišejo se vse razen zadnjih dveh verzij datotek tipa .OBJ in .MAC. Izpišejo se vse brisane datoteke.

3.4.6 PREIMENOVANJE DATOTEK /RE

S kretnico /RE spremenimo ime datoteke. Lahko uporabimo tudi podkretnico (/NV) s katero dosežemo, da ima preimenovana datoteka številko verzije za eno večjo od zadnje verzije datoteke, kateri spreminjamo ime. Kretnico /RE uporabljamo na naslednji način:

izh-dat=vh-dat1[,vh-dat2,,,,vh-datN]/RE[/NV]

izh-dat To je novo ime, ki bo dano datoteki. Specifikacija izhodne datoteke ima to posebno lastnost, da se za ime, tip in verzijo datoteke dovoljujejo eksplicitne vrednosti, zvezdice ali privzete vrednosti. UFD, ime datoteke, tip datoteke ali verzija, ki so bodisi zvezdice, ali privzete vrednosti, pomenijo, da se uporablja ustrezno polje vhodne datoteke. To pomeni, da kretnica /RE lahko spremeni enega ali več polj, ostala pa ohrani. Ime izhodne datoteke ima splošno obliko:

enota:[s,č]ime.tip;verzija

vh-datI To je ime datoteke, ki jo bomo preimenovali. Specifikacije vhodne datoteke so standardne in dovoljujejo uporabo zvezdic v vseh poljih, vključno UFD. Oblika zapisa vhodne datoteke je naslednja:

enota:[s,č]ime.tip;verzija

če ni imena, tipa ali verzije datoteke, se privzame vrednost *.*;*.

Kretnica /RE ne povzroči prenašanja podatkov. Staro ime datoteke se briše in vpiše se novo. Lahko prenesemo ime z enega direktorija v drugi. Direktoriji morajo biti na isti enoti zato, ker se podatki ne prenašajo.

/RE kretnica za preimenovanje

/NV podkretnica za novo verzijo. Ta podkretnica poveča številko verzije preimenovane datoteke za eno, če je potrebno ali da datoteki verzijo 1, če tako ime in tip še ne obstajata na direktoriju. Če podkretnico navedemo

na levi strani enačaja, velja za vse datoteke, in številke verzij se povečajo za 1 pri vseh obstoječih imenih datotek. Če navedemo podkretnico na levi, velja samo za tisto datoteko, za katero je navedena.

Primeri:

1. PIP>TEST.DAT;1=PODAT.DAT;5/RE
   Preimenuje se datoteka PODAT.DAT;5 v datoteko TEST.DAT;1.

2. PIP>TEMP.*;*=TMP1.1;*,TMP2.2*; /RE
   Preimenujejo se vse verzije datotek z imeni TMP1 in TMP2 v TEMP, pri tem pa se ohranja tip in verzija vsake datoteke.

4. PIP>[200,220]=[200,200]/RE
   Preimenujejo se vse datoteke z [200,200] na [200,220] in se pri tem ohrani ime, tip in verzija vsake datoteke.

3.4.7 SPREMINJANJE ZAŠČITE

S kretnico /Pr in ustreznimi podkretnicami spremenimo zaščito datoteke. Uporabniki so z obzirom na lastnika datoteke razdeljeni v štiri skupine: privilisirani (system), lastnik, skupina in ostali. Zaščita je definirana tako da za vsako skupino predpišemo kaj sme delati. Datoteko lahko beremo (R), pišemo (W), jo širimo (E) in brišemo (D). Zaščito si lahko ogledamo z /FU kretnico. Zaščito spremenimo s kretnico /Pr in podkretnicami /SY, /OW, /GR in /WO. zaščito lahko spremenita lastnik in privilisirani uporabnik.

Kretnico /PR uporabljamo na naslednji način:

PIP>ime-dat/PR[/SY:koda][/OW:koda][/GR:koda][/WO:koda][/FO]

kjer so:

ime-dat Ime, tip in verzija datoteke, katere zaščito spreminjamo; Specifikacija datoteke je obvezna, saj če je ni, potem privzeta vrednost ni *.*;*.

/PR slavna kretnica

/SY,/OW Podkretnice,s katerimi določamo spremembe zaščite.
/GR,/WO navedemo samo tiste podkretnice za skupine uporabnikov, katerim želimo spremeniti pravice.

koda koda določa novo zaščito naspram določene skupine uporabnikov. Kodo sestavljajo največ štiri črke R, W, E in D (za čitanje, pisanje, razširjanje in brisanje), ki predpišejo nove pristopne pravice za skupino, za ka-

tero podkretnico jo navedemo. Dejavnost, ki je ne označimo, ta skupina ne sme izvajati.

/FO S to podkretnico določimo za lastnika datoteke tisti UIC, na katerega direktoriju se datoteka nahaja. S tem smo spremenili hkrati pravice ostalih uporabnikov do te datoteke.

```
PIP>[100,50]*.*;*/PR/FO
```

določimo [100,50] za lastnika vseh datotek na direktoriju [100,50]

Primeri:

1. PIP>TEST.DAT;5/PR/OW:RWE/GR:RWE/WO
   Postavimo tako zaščito datoteke, da imajo lastnik in skupina vse vrste dostopov, razen možnosti brisanja; ostali nimajo nobenega dostopa. Sistemske zaščite so nespremenjene.

2. PIP>DE1;[*,*]*.*;*/PR/FO
   S tem povzročimo, da so lastniki datotek tisti UIC, s katerega UFD-ja so datoteke vnešene.

3.4.8 KONTROLNE KRETNICE

3.4.8.1 Kretnica /DF

S to kretnico spremenimo izbrano enoto in/ali direktorij. Običajno je izbrana (privzeta) enota za PIP SY0:, izbran (privzet) direktorij pa tisti UIC, pod katerim PIP trenutno deluje. Kretnica /DF spremeni le UFD (direktorij) in ne vpliva na UIC, ne na zaščito datotek. Ko iztopimo iz PIP-a, se vrnemo na prejšnji direktorij oz enoto. Kretnico /DF uporabimo na naslednje načine:

```
enota:[s,š]/DF
ali
enota:/DF
ali
[s,š]/DF
```

kjer so:

enota: Nova enota, na katero se bodo nanašali naslednji ukazi PIP. Enoto je moramo specificirati, če ne navedmo direktorij.

[s,š] Novi direktorij, na katerem se bodo izvajali naslednji

PIP ukazi. Če ni navedena enota, je obvezna specifikacija direktorija.

/DF Kretnica za privzemanje novih izbranih vrednosti.

Primeri:

1. PIP>[30,30]/DF
Postavi se nov izbrani direktorij [30,30]

2. PIP>DE1:/DF
Postavi se nova izbrana enota DE1:

3. PIP>DE1:[30,30]/DF
Postavi se nova izbrana enota DE1: in direktorij [30,30].

3.4.8.2 Previjanje traku

S kretnico /RW previjemo magnetni trak na začetek. Lahko ga uporabimo na vhodni ali izhodni strani. Kadar ga uporabimo na izhodni strani, se trak briše. To tehniko s pridom uporabimo za brisanje traku, predno pišemo nanje. Kretnica /RW uporabljamo na naslednji način:

```
PIP>izh-dat/RW=vh-dat
        ali
izh-dat=vh-dat/RW
```

kjer so:

izh-dat specifikacija izhodne datoteke

vh-dat specifikacija vhodne datoteke

/RW kretnica za previjanje. Kadar uporabimo kretnico /RW na vhodni strani, se trak previje pred odpiranjem vhodne datoteke. Če kretnice ne navedemo, procesor za magnetni trak išče datoteko do konca traku in če je ne najde, previje na začetek in išče do mesta, kjer je bil ukaz izdan. Če vemo, da se vhodna datoteka nahaja pred trenutno pozicijo čitalne glave, z /RW skrajšamo čas iskanja (,ki je lahko res dolgo).

3.4.8.3 Deblokiranje datoteke - kretnica /UN

Če je datoteka nepravilno zaprta (v izpisu PIP/LI se pred datumom kreiranja pojavi črka L), ji ne moremo pristopiti. Najprej jo moramo pravilno zapreti. To naredimo z kretnico /UN. Ko je datoteka pravilno zaprta, lahko ugotovite stanje datoteke in naredite potrebne korake za odpravo nepravilnosti. Kretnico /UN uporabimo na sledeči način:

PIP>datoteka1[,datoteka2,....]/UN

/UN se nanaša na vse navedene datoteke. Uporabimo lahko zvezdice v specifikaciji datoteke. Moramo biti lastnik datoteke ali pa privilisirani uporabnik na sistemu. Ime vsaj ene datoteke moramo navesti, ker ni privzetih vrednosti.

Primer:

PIP>DE1:[120,2]TEST1.DAT;2/UN

Datoteka TEST1.DAT;2 se pravilno zapre.

V A J E

1. Z EDT-jem kreirajte datoteko PRIMER1.FTN z vsebino:

```
    TYPE *,'PRIMER FORTRANSKEGA PROGRAMA'
    TYPE *,'PROGRAM RACUNA STEVILO PI TAKO DA'
    TYPE *,'RACUNA PLOSCINO VCRTANEGA IN OCRTANEGA MNOGOKOTNIKA'
    TYPE *,'STEVILO STRANIC POVECUJE DOKLER SE PLOSCINI NE '
    TYPE *,'RAZLIKUJETA ZA MANJ KOT 10**4'
    TYPE *,'VNESI POLMER KROGA'
    N=3
    PI=4*ATAN(1.)
10      N=N*2
    PN=N*R*SIN(PI/N)*R*COS(PI/N)
    Y=SIN(PI/N)/COS(PI/N)
    PZ=N*R*Y*R
```

2. Za vrstico TYPE *,'VNESI POLMER KROGA' vnesite vrstico

```
    ACCEPT *,R
```

3. Namesto vrstic

```
    Y=SIN(PI/N)/COS(PI/N)
    PZ=N*R*Y*R
```

vstavite vrstico:

```
    PZ=N*R**2*SIN(PI/N)/COS(PI/N)
```

4. Z EDT kreirajte datoteko FORTRAN2.FTN z vsebino:

```
    TYPE *,PZ,PN
    IF (PZ-PN-0.0001) 20,20,10
20  PI1=PN/R/R
    TYPE *,'STEVILO PI=',PI1
    END
```

5. Z EDT kreirajte datoteko PASKAL.PAS z vsebino:

```
PROGRAM RAST;
(* PROGRAM ZA IZRACUN RASTI PREBIVALSTVA *)
    VAR LETO, I :  INTEGER;
        STEV, RAST :  REAL;
BEGIN
```

DELO Z DATOTEKAMI
VAJE

```
      WRITELN('OD ZACETNEGA LETA, KI GA VNESEMO PREKO TERMINALA');
      WRITELN('VNESEMO TUDI ZACETNO STEVILO PREBIVALCEV IN STOPNJO
      WRITELN('RASTI KI NAJ BO >1 ');
      WRITE ('Vtipkajte zacetno leto:');
      READ(LETO);
      WRITE ('Vtipkajte stevilo prebivalstva v tem letu:  ');
      READ(STEV);
      WRITE ('Vtipkajte stopno rasti prebivalstva v procentih:  ')
      READ(RAST);
          I :=1;
      REPEAT
          LETO :=LETO+1;
          I :=I+1;
          STEV := STEV * (1+RAST/100);
      UNTIL I > 25;
      WRITELN('V LETU ',LETO,' BO PREBVALSTVO NARASLO NA ',STEV,'
   PREBIVALCEV');
  END.
```

6. Za 2. vrstico programa vnesite vrstice:

```
(* Pomen variabel:  LETO ...  leto stetja prebivalstva *)
(* - - - - - - - - - STEV ...  stevilo preb.  v letu LETO *)
(* - - - - - - - - - RAST ...  indeks rasti preb., ki je >1 *)
```

7. Za vrstico BEGIN vtipkajte vrstico

```
      WRITELN('PROGRAM IZRACUNA RAST PREBIVALSTVA V 25 LETIH');
```

8. Izpišite svoj direktorij na ekran. Izpišite samo datoteke tipa .FTN.

9. Naredite prejšnjo vajo z kretnico /FU

10. Združite datoteki FORTRAN1.FTN in FORTRAN2.FTN v datoteko AFORTRAN.FTN

11. Preverite zaščito datoteke AFORTRAN.FTN in spremenite zaščito tako, da bodo privilisirani (sistemski) uporabniki delali z datoteko vse razen brisanja, člani grupe samo brali, za ostale uporabnike pa naj bo datoteka nedostopna.

12. Izpišite direktorij in preverite novo zaščito datoteke AFOR-

TRAN.FTN

13. Prepišite datoteko AFORTRAN.FTN na datoteko BFORTRAN.FTN

14. Preimenujte datoteko BFORTRAN.FTN v FORTRAN.FTN in ponovno izpišite direktorij.

15. Izbrišite vse stare verzije vaših datotek in datoteke AFORTRAN in BFORTRAN

16. Izpišite na svoj terminal datoteko FORTRAN.FTN.

```
      WRITELN('OD ZACETNEGA LETA, KI GA VNESEMO PREKO TERMINALA')
      WRITELN('VNESEMO TUDI ZACETNO STEVILO PREBIVALCEV IN STOPNJO')
      WRITELN('RASTI KI NAJ BO >1 ');
      READ(LETO);
      WRITE ('Vtipkajte stevilo prebivalstva v tem letu! ');
      READ(STEV);
      WRITE ('Vtipkajte stopnjo rasti prebivalstva ');
      READ(RAST);
      I :=1;
      REPEAT
        LETO :=LETO+1;
        I :=I+1;
        STEV := STEV + (1+RAST/100);
      UNTIL I > 25;
      WRITELN('V LETU ',LETO,' BO PREBIVALSTVO NARASLO NA ',STEV,
       PREBIVALCEV');
    END.
```

6. Za 2. vrstico programa vnesite vrstico:

```
(* Pomen variabel: LETO ...  leto stetja prebivalstva *)
(* - - - - - - - - - STEV ...  stevilo preb.  v letu LETO *)
(* - - - - - - - - - RAST ...  indeks rasti preb., ki je >1 *)
```

7. Za vrstico BEGIN vtipkajte vrstico:

```
    WRITELN('PROGRAM IZRACUNA RAST PREBIVALSTVA V 25 LETIH');
```

8. Izpisite svoj direktorij na ekran. Izpisite samo datoteke tipa .FTN.

9. Naredite prejsnjo vajo z kretnico /FU

10. Združite datoteki FORTRAN1.FTN in FORTRAN2.FTN v datoteko AFORTRAN.FTN

11. Preverite zaščito datoteke AFORTRAN.FTN in spremenite zaščito tako, da bodo privilegirani (sistemski) uporabniki delali z datoteko vse razen brisanja, ostali drugi samo branje, za ostale uporabnike pa naj bo datoteka nedostopna.

12. Izpišite direktorij in preverite novo zaščito datoteke AFOR-

POGLAVJE4

DELO Z ENOTAMI IN NOSILCI PODATKOV

4.1 ENOTE SISTEMOV DELTA

Na sisteme DELTA lahko vežemo vrsto različnih perifernih enot: diskovne, DECtape, magnetnotračne, disketne, kasetne enote, čitalce in luknjalnike papirnatega traku, tiskalnike, zaslonske terminale, pisalnike, A/D pretvornike, laboratorijske periferne sisteme ... Vse periferne enote imajo poleg svojih hardwareskih oznak tudi imena, ki jih uporabljamo, ko izdajamo kakršne koli ukaze, ki se nanašajo na te enote. Ime enote se sestoji iz dveh ASCII znakov, ki označujeta tip enote, ena- ali dvomestnega osmiškega števila, ki pomeni zaporedno številko enote z istega tipa v našem sistem, ter znaka ":". Primer: DK2: . V tabeli so navedene najbolj pogoste vhodno/izhodne enote na sistemih DELTA/M.

vhodno/izhodna enota	ime enote
Diski	
RK05	DKnn:
RK06/07	DMnn:
RL01	DLnn:
RM02/03 diski	DRnn:
RS03/04 diski	DSnn:
RF diski	DFnn:
RP04/05/06	DBnn:
RP02/03 diski	DPnn:
RX01 diskete	DXnn:
RX02 diskete	DYnn:
Magnetni trakovi	
TU45/TU16/TE16/TU77	MMnn:
TS03/TU10/TE10	MTnn:
TS04	MSnn:
DECtrak (TC11)	DTnn:
DECtrak II (TU58)	DDnn:
Kasete	CTnn:
Vrstični tiskalnik	LPnn:
Luknjalnik papirnega traku	PPnn:
čitalec papirnega traku	PRnn:
Terminal	TTnn:

Tabela najbolj pogostih vhodno/izhodnih enot

4.1.1 PSEUDO ENOTE

Poznamo pa tudi pseudo ali lažna imena enot. Pseudo imena enot se morajo nanašati na resnične fizične enote. Zvezo med pseudo in imenom vzpostavimo z ASN ali pa RED MCR ukazom. Pseudo imena enot nam poenostavljajo naslavljanje enot.

Pseudo enota	ime
Konzolni tiskalnik	CL:
Konzolni izhod	CO:
Sistemska enota	LB:
Pseudo vhodni terminal	TI:
Uporabnikova privzeta enota	SY:
Null enota	NL:

Tabela pseudo enot

CL: je navadno vrstični tiskalnik, vendar je za task vseeno, ka-

teri tiskalnik (ali celo terminal) je to. Izpis se vrši na enoti, ki ima pridruženo pseudo ime CL:. CO: je navadno glavni operaterski terminal. LB: je diskovna enota, na kateri je operacijski sistem in sistemske knjižnice. TI: je uporabnikov terminal, preko katerega je prijavljen sistemu. SY: je uporabnikova privzeta enota. To je tista enota, h kateri pristopa sistem, ko zahtevamo pristop k datoteki in enote nismo navedli. (Sporočilo, katera je naša sistemska enota, dobimo z ukazom ASN.) NL: je pseudo enota, ki ni pridružena nobeni fizični enoti. Uporabljamo jo pri testiranju programov kot vhodno/izhodno enoto, kamor program spravlja podatke oz. kjer jih čita. Teh podatkov pa vresnici sploh ni. Testira se samo mehanizem delovanja programa.

4.1.2 LOGIČNA IMENA ENOT

Z uvajanjem logičnih imen enot dosežemo neodvisnost programov od konfiguracije računalnika. Logično ime se sestoji iz dveh ASCII znakov, eno- ali dvomestnega osmiškega števila ter dvopičja (:). Preden izvajamo program, moramo povezati logična imena enot, ki jih uporabljamo v programu, z imeni dejanskih enot. Povežemo jih z MCR ukazom ASN. Primer: če smo v programu uporabili logično ime XY: za disk, moramo to ime povezati z imenom dejanskega diska.(>ASN DL1:=XY:)

Povezava med logičnim in dejanskim imenom enote je lahko: lokalna, globalna in vključitvena.

lokalna velja samo za taske, katerih izvajanje smo sprožili z istega terminala kot smo izdali ukaz, ki je povezal logično in dejansko ime.

globalna velja za vse taske v sistemu.

vključitvena ko se vključimo v večuporabniški sistem, nam sistem avtomatsko izvede eno ali več povezav. Vključitvene povezave so močnejše od globalnih, lokalne pa lahko spremenijo vključitvene. (Ob vljučitvi nam sistem priredi eno od diskovnih enot kot našo sistemsko SY: in naš terminal kot TI: .)

4.1.3 OBČE, ZASEBNE IN BREZLASTNIŠKE ENOTE

V večuporabniških sistemih imamo lahko tri vrste enot glede na lastništvo: obče, zasebne in brezlastniške enote.

* obče enote lahko uporablja vsak v sistem vključen uporabnik. Privilisirani uporabnik lahko naravna enoto kot obče uporabno. Vsi uporabniki lahko potem pristopajo k taki enoti. Tako enoto ni mogoče alocirati.

* zasebna enota je taka, ki jo je uporabnik alociral zase. Pristopati sme sam im privilisirani uporabnik. Privilisirani uporabnik mora pred pristopom izdati MOUNT ukaz.
* Brezlastniška enota je taka, ki ni obča niti zasebna. če uporabnik (privilisiran ali neprivilisiran) izda MOUNT ukaz za to enoto, jo lahko uporabljajo tudi ostali uporabniki, če so izdali MOUNT ukaz.

Običajno so sistemski disk in delovni disk obči (PUBLIC) enoti, kar pomeni da lahko vsak uporabnik dela z njimi. če je enota brezlastniška, jo lahko setiramo kot občo z ukazom SET /PUB=enota:. Ta MCR ukaz je privilisiran. Nasprotni ukaz je SET /NOPUB=enota:. če pa želimo, da brezlasniška enota postane zasebna, izdamo ukaz ALLocate. nasprotni ukaz je DEAllocate. Občo enoto ne moremo direktno prisvojiti. Za določene operacije moramo prisvijiti enoto, za druge je to priporočljivo.

4.2 UKAZI ZA DELO Z ENOTAMI

4.2.1 ENOTE - DEVICES

kaz Devices prikaže na terminalu, s katerega je izdan, imbolična imena vseh enot, odnosno določenega tipa enot, ki jih istem pozna. Imena enot se pojavijo v prvem stolpcu, v drugem in ostalih stolpcih pa se po potrebi izpišejo informacije za vsako enoto.

blika ukaza:

```
DEVices [dd:]
DEVices [/LOG]
```

jer so:

dd: Tip enote, o kateri želimo dobiti informacije

/LOG Izpišejo se vsi terminali, na katerih so prijavljeni uporabniki

rimeri:

```
>DEV DL:
DL0:    PUBLIC MOUNTED LOADED TYPE=RL01
DL1:    PUBLIC MOUNTED LOADED TYPE=RL01

>DEV/LOG
TT2:    [200,7] - LOGID ON LOADED
TT4:    [50,6] - LOGID ON LOADED
TT12:   [20,70] - LOGID ON LOADED
TT14:   [50,7] - LOGID ON LOADED
TT17:   [50,5] - LOGID ON LOADED
```

4.2.2 UKAZA SET /PUB IN SET /NOPUB

Z ukazom SET /PUB setiramo enoto kot public, tj. občo. Vsak uporabnik jo sme lokalno pridružiti. Če v ukazu ne navedemo enote se izpišejo imena občih enot v sistemu. Neprivilisirani uporabnik ne more setirati enote kot obče.

```
/PUB[=enota:]
```

Primer:

```
>SET /PUB
PUB=DL0:
PUB=DL1:
PUB=DK0:
PUB=DK2:
PUB=DK3:
```

SET /NOPUB ima ravno nasprotni učinek kot SET /PUB. S tem ukazom lahko neko enoto naravnamo kot brezlastniško. Enota mora biti odstavljena. Če enote ne navedemo, dobimo na zaslonu izpisane vse enote, ki so privatne. Ukaz ima sledečo obliko:

```
SET /NOPUB[=enota:]
```

Primer:

```
>SET /NOPUB
NOPUB=DK1:
```

4.2.3 DODELITI - ALLOCATE

Z ukazom Allocate označimo določeno enoto za uporabnikovo zasebno enoto. (Ta ukaz se uporablja le na sistemih, ki podpirajo večuporabniško zaščito.) Dodelitev enote onemogoča nepriviligiranim uporabnikom dostop do enote. Le lastnik enote ali priviligirani uporabnik lahko uporablja ali sprosti zasebno enoto. Montirane enote, skupne enote ali druge zasebne enote uporabnikov ni možno dodeliti.

Sistem avtomatično sprosti zasebne enote uporabnika, ko se ta odjavi (izda ukaz Bye).

Oblika ukaza:

ALLocate dd[nn:][=llnn:]

Kjer so:

dd Mnemonično ime za enoto, katero želimo dodeliti

nn: številka enote, katero želimo dodeliti. V kolikor nn: ne uporabimo v ukazu, sistem avtomatično dodeli prvo prosto enoto tipa dd in izpiše dodeljeno enoto v naslednji obliki:

ALL--ddnn: NOW ALLOCATED

llnn: Ime logične enote, katero sistem kreira in priredi dodeljeni fizični enoti. (To je logična prireditev, kot pri ukazu ASN).

Primeri:

```
>ALL DK2:
```

Dodeli se disk DK2:

```
>ALL DK:
```

Dodeli se disk DK0:

```
>ALL DK=XX0:
 ALL--DK3: NOW ALLOCATED
```

Dodeli se prva prosta logična enota tipa DK (v tem primeru disk DK3:, ker so bile enote 0,1 in 2 že dodeljene prej ali pa skupna) in ji priredi logično ime XX0:.

4.2.4 DEALOCIRATI - DEALOCATE

Ukaz Deallocate sprosti zasebno (dodeljeno) enoto in s tem omogoči tudi ostalim uporabnikom dostop do enote. Neprivilisirani uporabnik lahko sprosti le tiste enote, katere si je sam dodelil. Če se lastnik zasebne enote odjavi sistemu (izda ukaz BYE) sistem avtomatično sprosti vse njegove zasebne enote.

Če je dodeljena enota montirana, jo je potrebno pred sprostitvijo dismontirati.

Oblika ukaza:

DEAllocate [ddn:]

kjer je:

ddn: Ime in številka zasebne (dodeljene) enote, ki naj se sprosti. Če v ukazu enote ne navedete, sistem sprosti vse zasebne enote uporabnika, ki je prijavljen na terminalu, s katerega je ukaz izdan.

Primer:

>DEA DK:

Sprosti se disk DK:. Sproščeno enoto lahko sedaj montirajo, ali si jo dodelijo ostali uporabniki.

4.2.5 PRIREDITI - ASSIGN

kaz Assign definira, briše ali prikaže prireditve logičnih enot. rirejanje logičnih enot povezuje logična imena s fizičnimi, seudo ali ostalimi logičnimi enotami. Ko uporabnik priredi ogični ali pseudo enoti logično ime, sistem sklene prireditev na rirejeno fizično enoto. Razlikujemo tri vrste logične prireditve enot: globalno, lokalno in prijavno.

- Globalne prireditve se nanašajo na vse programe, ki se v sistemu izvajajo. Samo privilisirani uporabniki lahko definirajo ali brišejo globalne prireditve.

- Lokalne prireditve se nanažajo le na programe, ki smo jih sprožili s terminala, s katerega so bile izdane prireditve. Vsak terminal lahko definira ali briše svoje lokalne prireditve.

- Prijavne prireditve se javljajo le v sistemih, ki podpirajo večuporabniško zaščito. Ko se uporabnik z ukazom Hello prijavi sistemu, sistem avtomatično priredi logično ime SY: tisti enoti, na kateri ima uporabnik svoje datoteke. Prijavne prireditve se nanašajo le na programe startane s terminala, kjer je trenutni uporabnik prijavljen. Samo privilisiran uporabnik lahko definira ali briše prijavne prireditve.

okalne prireditve imajo prednost pred prijavnimi in globalnimi rireditvami, prijavne prireditve pa imajo prednost pred globalnimi. Ko uporabnik briše lokalno logično ime enote, veljajo sistemske privzete vrednosti v primeru nasprotujočih si prijavnih rireditev.

Neprivilisirani uporabnik lahko naredi samo lokalne povezave. Neprivilisirane oblike ukaza:

ASN ppnn:=llnn

Priredi se logično ime llnn: enoti ppnn:. Ukaz postavi lokalno prireditev za terminal, s katerega je bil izdan.

ASN

Prikaže na terminalu, s katerega je bil izdan, vse lokalne in prijavne prireditve logičnih enot.

ASN =llnn:

Briše vse lokalne prireditve specificiranega logičnega imena enote, ali v kolikor je ukaz brez llnn: briše vse lokalne logične prireditve za terminal, s katerega je bil izdan.

Pomen kratic:

pp - fizično, logično ali psevdo ime enote
nn - številka
ll - logično ime enote
tt - ime terminala

Primer:

```
>ASN DE:=MD2:                !Definicija MD2:
>ASN MM:=MD1:                !Definicija MD1:
>ASN                         !Izpis lokalnih in prijavnih prireditev
MD2:     DE0:     LOCAL      TI - TTG:
MD1:     MM0:     LOCAL      TI - TTG:
SY:      DE0:     LOGIN      TI - TTG:
>ASN=MD1:                    !Brisanje lokalnega logičnega
>                            !imena enote MD1:
>ASN=MD2:                    !Brisanje lokalnega logičnega
>                            !imena enote MD2:
>ASN DK:=YY:                 !Prireditev YY: k DK:
>ASN YY:=ZZ:                 !Prireditev ZZ: k YY:
>ASN                         !Izpis prireditev
ZZ0:     DK0:     LOCAL      TI - TTG:
YY0:     DK0:     LOCAL      TI - TTG:
```

4.2.6 PRISTAVITI - MOUNT

Ukaz MOUNT vpostavi zvezo med fizično enoto in sistemskim sofwarom za vzdrževanje podatkovnih struktur. Za vsako enoto, za katero smo izdali ukaz MOUNT, se kreira 'file system control blok', ki ga ANCILLARY CONTROL PROCESSOR (ACP) uporablja za pristop k podatkom na enoti. MOUNT torej logično poveže določeno enoto s sistemom.

Nekateri pomožni programi in MCR ukazi pristopajo ne-pristavljenim enotam. (INI, DSC, BRU ..). ANSI standard trakovi morajo biti pristavljeni.

Če je neka enota odstavljena, se tekoča V/I operacija lahko opravi, vendar ni možno pristopati nekim drugim datotekam. Vsak uporabnik lahko izda MOUNT ukaz, vendar je potem enota pristavljena samo lokalno za njega, za druge uporabnike pa je odstavljena.

Neprivilegirani uporabnik lahko pristavi:

- svoje privatne enote (tj. alocirane); takim enotam potem drugi uporabniki ne morejo pristopati.
- pristavljene obče uporabne enote. K tem enotam lahko pristopajo vsi uporabniki. Pristavljanje izvedemo le, da se prepričamo, da je enota logično vezana na sistem. Pred tem mora privilegiran uporabnik pristaviti enoto.
- vse enote, ki niso bile alocirane ali določene kot obče uporabne. Samo tisti uporabnik, ki je pristavil tako enoto, kahko pristopa k njej.

Vsak privilegirani uporabnik lahko pristavi katerokoli enoto z naključnim pristopom. Tračno enoto lahko pristavi le en uporabnik.

DEC tark ali pa Files-11 enoto lahko pristavimo le, če je v sistemu prisoten task F11ACP. (S kretnico /ACP lahko vedno aktiviramo kak drug ACP.)

Obrazec ukaza:

```
MOUnt ddn:[labela][/kretnica(e)]
```

Neprivilegirani uporabnik mora vedno navesti labelo, medtem ko jo privilegiran lahko preskoči s kretnico /OVR. Če ukaz presega 80 znakov, povezujemo vrstice z črtico (-), vendar ne sme imeti ukaz več kot 512 znakov.

Kretnice:

/DENS /DENS=gostota. Gostota zapisa (v bitih na inčo, bpi) za določene vrste enot. Za tračne enote je lahko vrednost 800 (ali LOW) ali 1600 (ali HIGH); za diskete je LOW ali HIGH. Privzate vrednost so: 800 za TE16, TU45,

TU77; 1600 za TS04 in HIGH za RX02.

/EXT /EXT=število-blokov. Kretnica definira število blokov, ki jih sistem alocira k datoteki vsakokrat, ko le ta porabi ves ji do sedaj alociran prostor. Privzeta vrednost: prečita jo iz home blocka .

/FPRO /FPRO=[privilisirani,lastnik,skupina,svet] Kretnica za privzeto zaščito datotek. Pristopna koda za vsak nivo zaščite se sestoji 4 možnih znakov: R=čitanje, W=pisanje, E=dodajanje, D=brisanje in *=privzato. Za vsak nivo predpišemo zaščito tako, da navedemo s pomočjo oznak kaj dovoljujemo.Privzata vrednost je [RWED,RWED,RWE,R]. Oklepaji so del sintakse.

/OVR Ta kretnica je privilisirana; ukaz Mount ne čita labele in lahko pristavimo enoto, če tudi ne poznamo labele.

/UIC /UIC=[uic] specificira uporabniško indentifikacijsko kodo v obliko [grupa,član]- oklepaji so del sintakse.

/VI Na terminal dobimo izpisano poročilo o mediju, ki smo ga pravkar pristavili. Za enote z naključnim pristopom ima poročilo sledečo obliko:

```
MOUNT ** VOLUME INFORMATIONS **
CLASS     = FILES-11 ali FOREIGN
DEVICE    = ddnn:
LABEL     = labela medija
UIC       = [grupa,član]
VOL PRO   = [privilisiran,lastnik,grupa,svet]
FILE PRO  = [privilisiran,lastnik,grupa,svet]
CHARAC    = [ATCH,DCF]
ACP NAME  = ime ACP-ja
```

Pri magnetnih trakovih v poročilu nimamo rubrike VOL PRO in CHARAC.

Primer:

```
>MOU DK3:SOLAVAJE/VI
```

Na terminal dobimo sledeče poročilo:

```
MOUNT ** VOLUME INFORMATIONS **
CLASS     = FILES-11
DEVICE    = DK3:
LABEL     = SOLAVEJE
UIC       = [1,1]
VOL PRO   = [RWED,RWED,RWED,R]
FILE PRO  = [RWED,RWED,RWED,R]
CHARAC    = [ ]
ACP NAME  = F11ACP
```

.2.7 ODSTAVITI - DISMOUNT

kaz Dismount zahteva, da sistem za delo z datotekami označi noto za odstavitev in sprosti kontrolne bloke. Novi dostopi do datotek na enoti niso možni. Po zaključitvi vseh operacij na datotekah sprosti ACP kontrolne bloke in s tem zaključi odstavitveni postopek.

e je enoto pristavil le en uporabnik bo ukaz Odstaviti označil, da je enota logično izključena. V DELTA/M operacijskem sistemu ni Pomožnega Kontrolnega Procesorja (ACP) za nepristavljene enote, zato imajo možnost dostopa do takih enot le nekateri programi (INI, DSC in BAD). Če je več uporabnikov pristavilo enoto je dostop do enote omogočen dokler jo vsi ne odstavijo. Ko Files-11 ali sistem za ANSI magnetne trakove končata odstavitveno operacijo, se izpiše sporočilo

*** enota: DISMOUNT COMPLETE

na konzolnem terminalu (CO:). Možna je časovna zakasnitev med izdajo ukaza in izpisom sporočila. Zakasnitev nastopi, če obstaja večje število vhodno/izhodnih zahtev in/ali če je več operacij na datotekah na enoti. Ko se odstavitev konča, se na terminalu s katerega je bil ukaz izdan izpiše ime enote katero je uporabnik odstavil in to brez čakanja na zaključitev vhodno/izhodnih operacij ali sprostitve datotek.

Neprivilegirani uporabniki lahko odstavijo le tiste enote, katere so sami pristavili. Privilegirani uporabnik lahko odstavi vsako pristavljeno enoto.

Oblika ukaza:

```
>DMO ddn:[ime][/kretnica(e)]
       ali
>DMO /USER[/kretnica(e)]
```

Kjer je:

ddn: enota, kjer se nahaja medij, ki ga odstavljamo ime ime ima v Files-11 sistemu lahko do 12 znakov za diske in do 6 znakov za magnetne trakove. Uporablja se kot kontrola ali odstavljamo pravi medij. Ime ni obvezno (v kolikor v ukazu ime medija ne navedemo se ime ne testira).

Kretnice:

DEV /DEV. Privilegirana kretnica, s katero odstavimo s specificirane enote vse uporabnike. Enoto ne sme zasedati ACP.

USER /USER. S to kretnico odstavimo vse medije, ki jih je uporabnik pristavil.

TERM /TERM=ttnn:. Privilisirana kretnica. S to kretnico odstavimo vse medije, katere je uporabnik na terminalu ttnn: pristavil.

Primeri:

>DMO DK1: TESTNI

Odstavi se disk DK1:. Če je le en uporabnik pristavil medij, se z ukazom medij odstavi. Če pa je več uporabnikov pristavilo medij ostane ta pristavljen za ostale uporabnike.

>DMO DK1:/DEV

Medij v disku Dk1: se logično izključi in sprosti vse kontrolne bloke.

>DMO /USER/TERM=TT17:

Odstavijo se vsi s terminala TT17: pristavljeni mediji.

4.2.8 INICIALIZACIJA - INITVOLUME

Z ukazom INI vzpostavimo Files-11 strukturo na disku, magnetnem traku ali disketi. Na diskih se s ukazom inicializira medij (uničijo se vse obstoječe datoteke), zapiše se "bootstrap" in "home" blok in zgradi se struktura direktorijev. Na magnetnih trakovih pa se z ukazom zapiše ime medija (po ANSI standardu) in se uničijo vse obstoječe datoteke.

Uporabnik lahko inicializira medij le na svojih zasebnih in dodeljenih enotah (tistih, ki si jih je sam dodelil). Izbira ustreznih parametrov v ukazu zahteva globoko poznavanje Files-11 strukture. (v IAS/RSX-11 I/O Operations Reference Manual je detaljno opisana Files-11 struktura za diske in ANSI magnetne trakove.)

Oblika ukaza:

```
INItvolume ddn:labela[/kretnica(e)]
```

Za razširitve ukaza čez več vrstic moramo začeti z INI> odzivnikom ter znakom "-", ki ga postavimo na konec vrstice, če želimo nadaljevati v naslednji vrstici. Skupna dolžina ukaza ne sme biti večja od 512. znakov.

Kretnice:

```
/BAD=[opcija]
/CHA=[značilnosti]
/DENS=izbira gostote
/EXT=število blokov
/FPRO=[sistem, lastnik, grupa, ostali]
/INDX=mesto index datoteke
/INF=začetna velikost index datoteke
/LRU=število kontrolnih blokov datotek v spominu
/MFX=število datotek na mediju
/PRO=[sistem, lastnik, grupa, ostali]
/UIC=[grupa, član]
/WIN=število kazalcev
/VI
(/DENS je edina kretnica, ki jo lahko uporabimo za
inicializacijo magnetnega traku)
```

Kjer je:

ddn: Ime in številka enote na kateri inicializiramo medij

labela Ime medija, ki ima lahko do dvanajst znakov za diske in do šest znakov za magnetne trakove. Ime moramo specificirati saj z njim identificiramo medij.

Od navedenih kretnic si bomo mi ogledali samo nekatere in sicer:

DENS /DENS=gostota. S kretnico /DENS izberemo gostoto zapisa (bitov na inčo ali BPI) magnetnega traku. Možen je za-

pis z enojno gostoto (800 BPI) ali dvojno gostoto (1600 BPI). Kretnica lahko zavzame sledeče vrednosti:

/DENS=800
/DENS=1600
/DENS=HIGH
/DENS=LOW

Privzeta vrednost: /DENS=800
Kretnico /DENS uporabljamo tudi za preverjanje gostote zapisa na disketi. Če navedemo kretnico /DENS, se vrednost testira, v nasprotnem primeru pa se privzame gostota medija.

EXT /EXT=število blokov. S kretnico /EXT definiramo število blokov, ki se dodajajo, ko se datoteka razširja. Privzeta vrednost: /EXT=5

FPRO /FPRO=[sistem, lastnik, grupa, ostali]. Kretnica /FPRO definira privzete vrednosti za zaščito vseh datotek na mediju katerega inicializiramo. zaščito definiramo tako, da za vsako kategorijo navedemo, kaj sme delati z datoteko.
Privzeta vrednost: /FPRO=[RWED,RWED,RWED,R]

PRO /PRO=[sistem, lastnik, grupa, ostali]. S to kretnico se definirajo vrste dostopa do enote. Za vsako grupo uporabnika navedemo dovoljene dejavnosti.
Privzeta vrednost: [RWED,RWED,RWED,RWED].

UIC /UIC=[grupa, član]. S to kretnico definiramo lastnika medija. Za grupo in člana so dovoljena števila od 1 do 377(8). Oglata oklepaja sta del zahtevane sintakse.

WIN WIN=število kazalcev. S kretnico definiramo število kazalcev za mapiranje, ki se dodelujo oknom datotek. Okno datoteke je sestavljeno iz števila kazalcev za mapiranje. Nahaja se v spominu, ko datoteka odpremo. Privzeta vrednost: /WIN=7.

VI /VI. S kretnico zahtevamo izpis vseh INI kretnic in vrednosti na terminalu s katerega je bil ukaz izdan.

Primer:

>INI DE1:SISTEM1/UIC=[6,2]/FPRO=[RWED,RWE,R,R]<CR>

Parametri za inicializacijo v zgornjem ukazu so:

DE1: - ime enote
SISTEM1 - ime medija
/UIC - koda uporabnika lastnika medija: grupa=6, član=2

/FPRO - privzete vrednosti za zaščito datotek in to

po skupah:
sistem: čitanje, pisanje, razširjanje, brisanje
lastnik: čitanje, pisanje, razširjanje
skupa: čitanje
ostali: čitanje

V A J E

1. Izpiši vsa imena vseh enot na računalniku.

2. Izpiši na ekran imena vseh terminalov in posebej vseh terminalov, na katerih je nekdo prijavljen.

3. Preveri, katera je tvoja privzeta (SY:) enota. Izberi in priredi neko drugo diskovno enoto na tem sistemu kot svojo privzeto enoto. Preveri novo povezavo in jo nato izbriši.

4. Svoji privzeti enoti priredi lokalno logično ime AB2: ter preveri povezavo.

5. Preveri, katere enote so obče (PUBLIC) in katere privatne (NOPUBLIC).

POGLAVJE5

RAZVOJ PROGRAMOV IN NJIHOVO IZVAJANJE

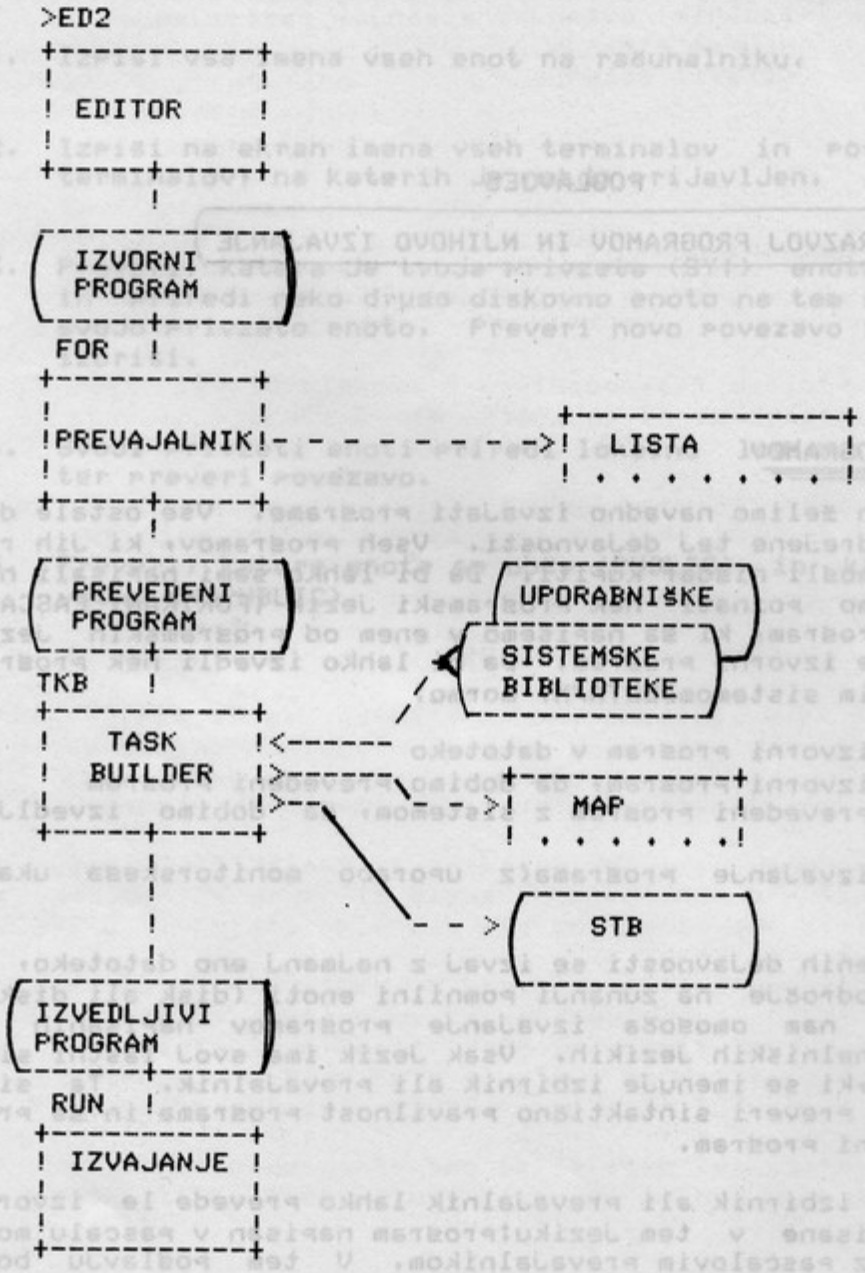
5.1 RAZVOJ PROGRAMOV

Na računalnikih želimo navadno izvajati programe. Vse ostale dejavnosti so podrejene tej dejavnosti. Vseh programov, ki jih rabimo, ne bomo mogli nikdar kupiti. Da bi lahko sami napisali nek program, moramo poznati nek programski jezik (FORTRAN, PASCAL, COBOL....). Program, ki ga napišemo v enem od programskih jezikov, se imenuje izvorni program. Da bi lahko izvedli nek program pod operacijskim sistemomDELTA/M, mormo:

- zapisati izvorni program v datoteko
- prevesti izvorni program, da dobimo prevedeni program
- povezati prevedeni program z sistemom, da dobimo izvedljiv program
- sprožiti izvajanje programa(z uporabo monitorskega ukaza RUN).

Vsaka od navedenih dejavnosti se izvaj z najmanj eno datoteko, ki predstavlja področje na zunanji pomnilni enoti (disk ali disketa). DELTA/M nam omogoča izvajanje programov napisanih v številnih računalniških jezikih. Vsak jezik ima svoj lastni sistemski program,ki se imenuje izbirnik ali prevajalnik. Ta sistemski program preveri sintaktično pravilnost programa in ga prevede v prevedeni program.

Vsak jezikovni izbirnik ali prevajalnik lahko prevede le izvorne programe, napisane v tem jeziku:program napisan v pascalu mora biti preveden z pascalovim prevajalnikom. V tem poglavju bomo prikazali razvoj fortranskega programa. Enako lahko naredimo s programi napisanimi v PASCALU, COBOLu ali drugih računalniških jezikih. Edina razlika je v prevajalniku, ki ga uporabljamo.


>ED2
EDITOR
IZVORNI PROGRAM
FOR
PREVAJALNIK
LISTA
PREVEDENI PROGRAM
UPORABNIŠKE SISTEMSKE BIBLIOTEKE
TKB
TASK BUILDER
MAP
STB
IZVEDLJIVI PROGRAM
RUN
IZVAJANJE

5.1.1 VNAŠANJE IZVORNEGA PROGRAMA

Izvorni program vnesemo s pomočjo enega od editorjev v tekstovno datoteko tako, da predstavlja ena vrsta v datoteki eno vrsto v programu. Pri tem se držimo vseh pravil določenega jezika. Mi bomo uporabili editor EDT, ki smo ga spoznali v tretjem poglavju. Ko kličemo EDT navedemo ime in tip datoteke, ki bo vsebovala izvorni program. Ime datoteke z izvornim programom nima nobene zveze z imenom programa kot jih poznajo nekateri prevajalniki (COBOL: PROGRAM-ID.imr-programa., ali v FORTRANU). Tip datoteke je v principu lahko kakršen koli, vendar operacijski sistem DELTA/M podpira nekatere standardne tipe za datoteke z izvornimi programi, uporaba katerih nam pri razvoju programa skrajšuje pisanje, hkrati pa dosežemo večjo enotnost pri delu v računalniškem centru.Na spodnji tabeli se nahaja nekaj standardnih tipov datotek za izvorne programe v različnih jezikih:

TIP	JEZIK
PAS	PASCAL
CBL	COBOL
BAS	BASIC
FTN	FORTRAN
MAC	MACRO

Kot primer bomo skozi vse poglavje razvijali program VSOTA, napisanega v FORTRANU. Ker je program FORTRANski, bo tipa .FTN in datoteka z izvornim programom se bo imenovala VSOTA.FTN. VSOTA.FTN zahteva vnos dveh realnih števil A in B preko terminala, ki ji potem sešteje in rezultat izpiše na terminal.

Naredimo torej prvi korak: pokličimo editor EDT in vnesimo program v datoteko VSOTA.FTN

```
>EDT VSOTA.FTN
Input file does not exist
[EOB]
*C<CR>
        TYPE *,'VNESI DVE ŠTEVILI A IN B'
        ACCEPT *,A,B
        C=A+B
        TYPE *,'VSOTA ŠTEVIL A IN B =',C
        CALL EXIT
        END
<GOLD><PAGE/COMMAND>
Command:EXIT<ENTER/SUBST>
>
```

5.1.2 PREVAJANJE IZVORNEGA PROGRAMA

Ko smo ustvarili izvorno datoteko (v FORTRANU, VSOTA.FTN v tem primeru,) sledi prevajanje programa v obliko, ki jo lahko bere sistem (binarna). Taka oblika programa se imenuje prevedeni pro-

sram ali object.

Prevajanje začnemo s tem, da pokličemo ustrezen prevajalnik, v našem primeru fortranski prevajalnik. To lahko naredimo na dva načina:

```
>FOR<CR>
FOR>ukazna vrsta
```

Najprej smo poklicali odzivni znak fortranskega prevajalnika in nato vnesli ukazno vrsto.

```
>FOR ukazna vrsta
```

Za besedo FOR takoj napišemo ukazno vrsto. Pogosteje uporabljamo drugi način. Včasih dobimo na ukaz FOR naslednje sporočilo:

```
MCR -- TASK NOT IN SISTEM
```

To pomeni, da fortranski prevajalnik ni v našem sistemu. Aktiviramo ga z naslednjim monitorskim ukazom:

```
>RUN $FOR<CR>
```

Ta ukaz pove sistemu naj vstavi fortranski prevajalnik v spomin in ga aktivira. Prevajalnik odgovori z odzivnim znakom:

```
FOR>
```

Za COBOL-ski ali pascalov prevaljalnik je postopek enak, le da besedo FOR zabenjamo z CBL ali PAS.

Za odzivnim znakom prevajalnika zapišemo ukazno vrsto. Splošna oblika ukaza za prevajanje fortranskega izvornega programa ima obliko:

```
FOR [ime1][,ime2]=ime3<CR>
```

ime1 prevedeno obliko izvornega programa, ki se nahaja v datoteki ime3.FTN, zapiše prevajalnik v datoteko ime1.OBJ. ime1 smemo izpustiti; v tem primeru ne dobimo prevedene oblike našega programa. Če smo navedli ime2, dobimo listo, če pa tudi tega nismo navedli, dobimo le kratko sporočilo o napakah. Fortranski prevajalnik poišče datoteko ime3.FTN na našem direktoriju, jo prečita ter jo prevede v binarno obliko. Prevedeno obliko izvornega programa spravi v datoteko ime1.OBJ na našem ditektoriju. V zgornjem ukazu smemo tip datoteke na obeh straneh izpustiti. Fortranski prevajalnik bo avtomatično poiskal datoteko tipa .FTN in imenom, ki ga navedemo in ravnotako kreiral datoteko tipa .OBJ z imenom, ki ga navedemo.

ime2 datoteka s tem imenom dobi tip .LST in v njo se vpiše lista, ki jo naredi prevajalnik. Lista prikaže sintak-

tične napake v programu in nekatere statistike. Če je aktiven QUEUE MANAGER, se lista takoj pošlje na tiskalnik. To lahko preprečimo z kretnico /-SP za ime2. Če namesto ime2 zapišemo TI: se lista izpiše na ekran, in je nimamo na disku. Ime2 smemo izpustiti. Če ime2 (oz. TI:) izpustimo, prevajalnik v primeru napak v našem izvornem programu izpiše kratko sporočilo o napakah, ki jih je odkril in koliko napak je odkril.

ime3 je ime datoteke z izvornim programom. Datoteka mora biti tipa .FTN; torej ime3.FTN in mora obstajati na našem ali kakem drugem direktoriju. Če datoteka ni tipa .FTN (oz. ustreznega tipa za določen prevajalnik) moramo tip datoteke eksplicitno navesti.

POMNI!
Izhod iz procesa je vedno na levi strani enačaja, vhod v proces pa na desni.

V praksi pišemo navadno tako, da je ime1=ime2=ime3 in potem po imenu vidimo,kateri prevedeni program (.OBJ) oz. lista (.LST) pripada nekemu izvornemu programu. Različna imena so dovoljena, ni pa to praktično.

Običajno uporabljamo standardne tipe datotek pri razvoju programov, kadarkoli je to mogoče! S tem si skrajšujemo pisanje ukazov.

Privzeti tipi datotek so prikazani na spodnji tabeli

Datoteka		Privzeti tip datoteke
Vhod	Izhod	
izvorna datoteka		.FOR (.CBL,.PAS)
	izpisna datoteka	.LST
	prevedeni program	.OBJ

Z dodatnimi kretnicami za vhodne in/ali izhodne datoteke lahko uvedemo posebne prevajalne zahteve. Vsi primeri v tem delu vsebujejo prevajalne privzete vrednosti.

Primeri:

Na primeru našega programa VSOTA.FTN bomo prikazali prevajanje fortranskega programa.

```
>FOR VSOTA=VSOTA<CR>
```

Ta ukaz pove prevajalniku, naj prevede izvorno datoteko VSOTA.FTN v prevedeni program imenovan VSOTA.OBJ. Dobimo samo prevedeni

program brez liste. Na ekranu se v primeru napak izpiše le kratko sporočilo.

```
>FOR VSOTA,VSOTA=VSOTA<CR>
```

Druga VSOTA na izhodni strani ukaza pove prevajalniku, da naj izvede tudi izpisno datoteko imenovano VSOTA.LST za izvorni program. VSOTA.LST se izpiše tudi na printerju.

```
>FOR ,VSOTA=VSOTA<CR>
```

Dobimo le listo na naš direktorij in na printer, prevedenega programa pa ne dobimo.

```
>FOR VSOTA,TI:=VSOTA<CR>
```

Lista se izpiše na našem zaslonu, prevedeno obliko pa dobimo kot datoteko na disku na našem direktoriju.

5.1.3 POVEZOVANJE PROGRAMA S SISTEMOM

DELTA/M program za povezovanje spremeni prevedeni program narejen z jezikovnim prevajalnikom v samostojni izvedljivi program. Ta izvedljivi program je datoteka tipa .TSK na disku. Izvajanje sprožimo z monitorskim ukazom RUN. Pri povezovanju povežemo en ali več objektov ter dodamo module iz ustreznih knjižnic.

Vse karakteristike taska (prioriteta, ime, ...) se določijo pri povezovanju. Program za povezovanje setira privzete vrednosti, če mi sami ne predpišemo drugače. Karakteristike taska lahko določimo z kretnicami ali pa z dodatki (opcijami). Nekaj jih bomo spoznali na koncu.

Popolna oblika ukaza za povezavo uporablja tri izhodne datoteke in več vhodnih datotek. Tri izhodne datoteke so:

1. datoteka,ki vsebuje izvedljiv program (tip datoteke TSK) in ki sa lahko instaliramo in izvajamo
2. datoteka ureditve spomina (MAP), ki vsebuje podatke o velikosti in legi različnih delov programa.
3. datoteka definicij simbolov (STB), ki vsebuje podatke o definicijah slobalnih simbolov.(Večini povezovalnikovih operacij STB datoteka ni potrebna.)

Vhodne datoteke vsebujejo vse prevedene programe in programe, ki so potrebni za tvorbo posameznega izvedljivega programa. Popolna oblika ukaza je:

```
>TKB ime1.TSK,ime2.MAP,ime3.STB=ime4.OBJ,LB:[ufd]knjižnica/LB
```

V tem ukazu smemo tipe datotek izpustiti, ker sistem sam dodeljuje izhodnim datotekam tega procesa standardne tipe datotek. Tudi vhodne datoteke imajo ustrezne tipe, če smo se ves čas

držali standardov.

Tabela prikazuje standardne tipe datotek v ukazu za povezavo

Datoteka		
Vhod	Izhod	Privzeti tipi dat.
Prevedena dat. knjiznica		.OBJ .OLB
	izvedljivi program dat.ureditve spomina dat.defin.simbolov	.TSK .MAP .STB

Tako lahko ukaz skrajšamo v naslednjo obliko:

>TKB [ime1][,ime2][,ime3]=ime4,LB:[ufd]knjižnica/LB

Za program VSOTA.FTN vnesemo naslednji ukaz:

>TKB VSOTA.TSK,VSOTA.MAP,VSOTA.STB=VSOTA.OBJ,LB:[1,1]FORLIB/LB

V tem ukazu je LB psevdonim za enoto, ki vsebuje knjižnico, [1,1] pa je UFD, kjer je knjižnica shranjena. FORLIB je fortranova sistemska knjižnica.

Z uporabo povezovalnikovih privzetih vrednosti lahko ukaz skrajšamo na:

>TKB VSOTA,VSOTA,VSOTA=VSOTA,LB:[1,1]FORLIB/LB<CR>

5.1.3.1 SKRAJŠANE OBLIKE UKAZOV

Kot prevajalnik nam tudi povezovalnik omogoča izpustitev nekaterih izhodnih datotek ali usmeritev izhodnih datotek na terminal. Če izpustimo datoteko v začetku ali v sredini seznama izhodnih datotek, moramo vstaviti vejico na mesto imena datoteke, da označimo prazno polje. Na vhodni strani ukaza, pa uporabljamo vejico, da ločimo eno vhodno datoteko od druge. Tabela prikazuje tipe datotek, ki nastanejo pri povezovanju programov.

Tabela tipov datotek, ki nastanejo pri povezovanju

Ukaz	Nastale izhodne datoteke
>TKB ime1,ime2,ime3=ime4	vse tri izhodne datoteke
>TKB,,ime3=ime4	samo STB datoteka
>TKB,ime2,ime3=ime4	MAP in STB datoteki
>TKB ime1,,ime3=ime4	TSK in STB datoteki
>TKB ime1,ime2=ime4	TSK in MAP datoteki
>TKB ime1=ime4	samo TSK datoteka

5.1.3.2 VEČVRSTIČNI POVEZOVALNIKOVI UKAZI

Veliko vhodnih datotek lahko povzroči, da je povezovalnikov ukaz daljši od ene vrstice. V tem primeru pokličemo pove zovalnik v naslednji obliki:

```
>TKB<CR>
TKB>
```

Ko smo vpisali <CR>, monitor aktivira povezovalnik, ki vrne TKB> odzivni znak. Povezovalnik izpiše ta znak po vsaki vrsti vnosa, dokler ne vpišete dve poševni črti (//) in <CR> v začetku vrste. Poševni črti končata delo povezovalnika in nas vrneta v MCR. Enovrstični ukaz

```
>TKB ime1,ime2,ime3=ime4,ime5,...,LB:[ufd]knjižnica/LB<CR>
```

lahko vnesemo kot

```
TKB<CR>
TKB>ime1,ime2,ime3=ime4<CR>
TKB>ime5,ime6,...,<CR>
TKB>LB:[ufd]knjižnica/LB
TKB>//<CR>
>
```

Do sedaj smo navajali samo eno knjižnico. Priključimo jih pa lahko tudi več in to tako, da jih navajamo eno za drugo. Ločimo jih z vejico. Knjižnice se lahko nahajajo na različnih diskovnih enotah.

5.1.3.3 POVEZOVALNIKOVA KRETNICE IN DODATKI

Povezovalnik ima na razpolago različne kretnice in dodatke, s katerimi kontroliramo nastanek izvedljivega programa oziroma določamo posebne lastnosti nastalega taska. Če jih ne navedemo, se privzemajo njihove privzete vrednosti.

5.1.3.3.1 Kretnice

Kretnice navajamo ob ustreznih vhodnih oziroma izhodnih datotekah povezovalnika. Za imenom ustrezne datoteke lahko navedemo več kretnic eno za drugo. Omenili bi samo nekaj kretnic:

/CP navajamo za imenom taska (ime1/CP) in specificira, da se sme task umikati (checkpointirati) iz pomnilnika. To je tudi privzeta vrednost. Da bi dosegli nasprotni učinek, torej neumakljivost pišemo kretnico /-CP.

/LB označuje, da je vhodna datoteka, za katero stoji, knjižnica. Če jo navedemo brez argumentov, pomeni, da se nedefinirani simboli poiščejo v knjižnici. Če navedemo argumente kretnice (/LB:mod1:mod2...modn), se samo navedeni moduli vključijo v task.

/PM navaja se za imenom taska in pomeni, da v primeru nasilne prekinitve izvajanja programa (ABO ukaz) sistem proizvede posnetek stanja taska v trenutku prekinitve. Če to ne želimo, pišemo kretnico /-PM. Privzeta vrednost je /-PM.

5.1.3.3.2 Dodatki

Dodatki ali opcije vnašamo posebej. Postopek je sledeči: najprej pokličemo TKB odzivni znak, vnesemo imena vseh izhodnih in vhodnih datotek ter ustrezne kretnice. Potem pa namesto dveh vnesemo eno poševno črto kot edini znak za TKB> odzivnim znakom. Sistem nam javi ENTER OPTIONS. Nato vnesemo željene opcije. Končamo z dvema poševnima črtama za TKB> odzivnim znakom.

```
>TKB<CR>
TKB>ime1,ime2,ime3=ime4,LB:[ufd]knjižnica/LB<CR>
TKB>/<CR>
ENTER OPTIONS:
TKB>opcija1<CR>
TKB>opcija2<CR>
   .
   .
   .
TKB>//<CR>
```

Opcij je veliko. Najdete jih lahko v knjigi Task Builder Manual.

5.2 STARTANJE IN KONTROLA IZVAJANJA PROGRAMOV

Program, ki smo ga razvili, je datoteka na disku in operacijski sistem ne ve, da je to task. Vsak program, ki ga želimo izvajati, mora biti instaliran - vpisan v STD - System Task Direktory. Instaliramo ga lahko začasno ali za stalno. Začasno instaliramo z MCR ukazom RUN, ko kot argument navedemo ime datoteke, v kateri se nahaja task. Začasno instalirani taski se brišejo iz STD, ko se konča njihovo izvajanje. Za stalno instaliramo task z privilegiranim MCR ukazom INStal. Spisek instaliranih taskov dobimo na ekran z ukazom TAS. Med trajno instaliranimi taski obstajajo večuporabniški taski. Obižajno so to pomožni programi, nekateri MCR ukazi, urejevalniki tekstov in prevajalniki. Imena teh taskov imajo v spisku instaliranih taskov (STD) na začetku tri pike in nato tri črke (recimo ...PIP, ...FOR). Večuporabniške taske lahko starta več uporabnikov istočasno. Prvi uporabik, ki ga starta, dobi original (recimo ...PIP), ostali pa dobijo kopije. Večuporabniške taske startamo tako, da vtipkamo tri črke iz imena, pod katerim je instaliran (recimo >PIP<CR>). Instalirane taske, ki niso večuporabniški, startamo tako z ukazom RUN, za katerim navedemo ime, pod katerim je task instaliran (vpisan v STD). Task se takoj izvede.

Sintaksa:

RUN ime-taska

Taski, ki obstajajo le kot datoteke na kakem disku in niso instalirani, startamo z ukazom RUN, za katerim navedemo ime datoteke, v kateri se nahaja task. Task se začasno instalira pod imenom TTn (n je številka našega terminala), izvede ter briše iz STD.

Sintaksa:

RUN [ddn:][[g,č]]ime[/kretnica(e)]

RUN $ime[/kretnica(e)]

kjer pomeni:

ddn: enota,na kateri se nahaja task datoteka. Privzeta vrednost: uporabnikova SY: enota.

[g,č] Direktorij, na katerem se nahaja task datoteka. Privzeta vrednost je trenutni uporabnikov UIC.

$ če namesto enote in UFD navedemo samo znak $, ukaz RUN poišče task na sistemskem disku (LB:) na direktoriju [1,54]. Na ta način navadno startamo pomožne programe, ki niso instalirani.

ime ime datoteke, v kateri se nahaja task, ki ga želimo izvesti. Datoteka mora biti tipa .TSK (če ni, moramo tip datoteke navesti) in povezana. Task se začasno instalira po imenom našega terminala.

Kretnice:

/CKP /CKP=opcija. Opcija je lahko YES, če naj bo task umakljiv (checkpointable) ali NO, če naj bo task neumakljiv (noncheckpointable). Privzeta vrednost je, kar smo navedli pri tvorbi taska kretnico /CP.

/PAR /PAR=ime-particije specificira ime particije, v kateri naj se task izvaja. Privzeta vrednost je, kar smo navedli pri tvorbi taska.

/PMD /PMD=opcija. Opcija je YES, če naj se ob nenormalni prekinitvi izvajanja taska naredi Post-Mortem Dump, in NO, če se Post-Mortem Dump naj ne naredi. Privzeta vrednost je tista, ki jo navedemo ob tvorbi taska s kretnico /PM.

/TASK /TASK=ime-taska. Ime-taska je ime, ki ga bo imel task za časa izvajanja v STD-ju. Privzeta vrednost je TTnn, kjer je nn številka našega terminala.

/UIC /UIC=[g,č]. UIC, pod katerim se bo task izvajal. Oklepaji so del sintakse. Privzeta vrednost je UIC, pod katerim je bil ukaz izdan.

Primeri:

```
>RUN PRIMER
```

če obstaja instaliran task PRIMER se bo izvedel ta, če ne, se poišče na našem direktoriju datoteka PRIMER.TSK in se izvede.

```
>RUN $MAC
```

Izvedel se bo task MAC.TSK s sistemskega direktorija. Task se bo avtomatsko instaliral, izvedel in umaknil iz STD-ja.

Kot primer si oglejmo še tri izvedbe našega programa VSOTA. Program je interaktiven, kar pomeni da podatke vnašamo preko terminala in rezultat se spet zapiše na terminal.

```
>RUN VSOTA
VNESI DVE ŠTEVILI A IN B
4,5 VSOTA ŠTEVIL A I B =9

>RUN VSOTA
VNESI DVE ŠTEVILI A IN B
12,15 VSOTA ŠTEVIL A I B =27

>RUN VSOTA
VNESI DVE ŠTEVILI A IN B
21.4,73.05 VSOTA ŠTEVIL A I B =94.45
```

5.2.1 PREKINITI - ABORT

Z ukazom ABOrt prekinemo izvajanje določenega programa. Neprivilisirani uporabnik lahko prekine le neprivilisiran program s svojega terminala. Privilisirani uporabnik lahko prekine vsak program.

Prekinitev programa sprozi zaustavljanje programa, ki je sestavljeno iz več korakov. V postopku zaustavljanja programa opravi operacijski sistem sledeče:

- Zaključi vhodno/izhodne operacije.
- Od rutine za označevanje zaključitve programa (Task Termination Notification routine - TKTN) zahteva izpis sporočila na terminalu, ki za prekinjeni program predstavlja TI:. Sporočilo je sestavljeno iz opisa vzroka za zaključitev in v kolikor je bil program povezan s /PM kretnico povezovalnika (Task Builder) odnosno instaliran s /PMD=YES kretnico tudi iz vrednosti registrov ob zaključitvi programa. V kolikor pa je bil program ob času izvajanja TKTN rutine izločen iz spomina (checkpointed), se vrednosti registrov ne izpišejo.
- Sprosti particijo v kateri se je program izvajal, če ni bil fiksiran.

Če ukaz vsebuje še kretnico /PMD, se ob prekinitvi izpišejo tudi vrednosti v registrih ne glede na to ali je bil program povezan s kretnico /PM ali instaliran s kretnico /PMD=YES. Ravno tako /-PMD blokira izpis.

Oblika ukaza:

ABOrt [ime-taska][,/PMD]

kjer so:

ime-taska Ime programa, ki ga želimo prekiniti. Če ime programa ne navedemo, se ukaz nanaša na program TTnn, kjer je nn oktalna stevilka terminala.

/PMD Kretnica, s katero zahtevamo izpis vsebine registrov ob prekinitvi programa.

Primeri:

```
>ABORT TEST
 TASK "TEST" TERMINATED
 ABORTED VIA DIRECTIVE OR MCR
      PC=
      PS=
      R0=
      R1=
      R2=
      R3=
```

```
        R4=
        R5=
        SP=

>ABO <CR>
 TASK 'TT7' TERMINATED
 ABORTED VIA DIRECTIVE
```

Z ukazom smo prekinili program TT7 s terminala TT7:.
Za prekinitev programa z imenom TTn ni potrebno navesti
v ukazu ime programa.

5.2.2 ZAUSTAVITI - STOP

Ukaz STOP zaustavi izvajanje taska TTnn oz. tistega, ki smo ga navedli (če se task seveda izvaja). Task se tudi več ne poteguje za prostor v pomnilniku. Neprivilegirani uporabnik lahko zaustavi izvajanje samo tistih taskov, ki so bili sproženi z istega terminala kot je izdan ukaz za zaustavljanje.

Oblika ukaza:

STP [ime-taska]

kjer je:

ime-taska je ime taska, katerega izvajanje želimo zaustaviti. Če imena taska ne navedemo, ukaz zaustavi task TTnn.

Nasprotni ukaz od STP je UNSTOP. Z ukazom UNS sprožimo izvajanje taska od tam, kjer je bil zaustavljen. Neprivilegirani uporabnik lahko nadaljuje izvajanje samo tistih taskov, katerih izvajanje je bilo sproženo z istega terminala.

Oblika ukaza:

UNS [ime-taska]

kjer je:

ime-taska je ime taska, katerega izvajanje želimo nadaljevati. Če imena taska ne navedemo, ukaz nadaljuje izvajanje taska TTnn (če je seveda bil zaustavljen).

5.2.3 AKTIVNI TASKI - ACTIVE

Ukaz ACTive prikaže na terminalu s katerega je bil izdan, imena s terminala sproženih programov. V kolikor je bil izdan s kretnico /ALL (ACT/ALL) prikaže na terminalu imena vseh aktivnih programov v sistemu.

Ukaz Active lahko izdamo tudi s kretnico /TERM (ACT /TERM=TTn:) da na TI: prikaže imena vseh aktivnih programov za specificirani terminal.

Oblika ukaza:

```
ACTive [/kretnica]
```

kretnici:

```
/ALL
/TERM=TTn:
```

Primeri:

```
>ACT
MCR...
...MCR
>ACT /ALL
...LDR.
F11ACP
MCR...
...MCR
...EDI
...KED
...FUT
...PIP
PIPT10
>ACT /TERM=TT20:
TT20
...EDI
```

5.2.4 SPISEK AKTIVNIH PROGRAMOV - ACTIVE TASK LIST

S tem ukazom se na terminalu, s katerega je izdan, prikažejo imena in stanja vseh aktivnih programov v sistemu, odnosno stanje željenega programa. Izpis vsebuje zadosti informacij za natančno določitev stanja vsakega aktivnega programa. Za vsak program se izpišejo naslednje informacije:

- Ime programa
- Fizični naslov (oktalno) kontrolnega bloka programa
- Ime particije
- Fizični naslov (oktalno) kontrolnega bloka particije
- Začetni in končni fizični naslov particije (oktalno)
- Statična in izvajalna prioritete programa
- Opisi statusa programa
- TI: kateri terminal - fizično ime
- število vhodno/izhodnih operacij (decimalno)
- Programu lokalni znanilci dogodkov
- Vrednosti v registrih in PSW
  (le za programe v spominu)

Prikazani opisi statusa programa izhajajo iz kontrolnega bloka programa (TCB). Imena, pred katerimi se nahaja znak minus (-) pomenijo, da ima status nasprotno vrednost. Na primer, -EXE pomeni, da se program ne izvaja. Tabela možnih statusov se nahaja v MCR Operations Manualu.

V kolikor program ni v spominu (v opisu se nahaja OUT) se vrednosti registrov, programskega števca in statusa ne izpišejo.

Oblika ukaza:

ATL [ime-taska]

Primer:

```
>ATL
 ..LDR|053626|LDR|  053572|00000000-00000000| PRI-248. DPRI-248|
   STATUS:  -CHK FXD STP PRV|
   TI - COO:| IOC - 0.|EFLG - 000001 000000 PS-170000 PC-042516
   REGS 0-6  000162 005620 177773 102320 000640 177777 045621|
MCR... 054222 SYSPAR 103572 00000000-00000000 PRI-99. DPRI-99.
   STATUS:  -PMD STP PRV NSD CAL
   TI - TT0: IOC - 0. EFLG - 000001 004000 PS-170000 PC-142516
   REGS 0-6  045162 235020 567773 100320 070640 177777 345621
        (operater je pritisnil CTRL/O in zaustavil izpis)
```

5.2.5 LISTA INSTALIRANIH TASKOV

Ukaz izpiše na zaslon listo instaliranih taskov. Izpis vsebuje:

1. ime taska
2. verzija taska
3. ime particije
4. prioriteta taska
5. dolžina taska v zlogih (osmiško)
6. oznaka enote, s katere se čita v pomnilnik
7. logični naslov prvega bloka na disku (osmiško)
8. stanje taska v pomnilniku

Oblika ukaza:

TAS

Primer izpisa na zaslon:

```
  1.      2.      3.     4.      5.       6.      7.        8.
------ ------ ------ ---- -------- ---- -------- -------
...LDR 07.05  LDRPAR 248. 00002200 LB0:-01035303 FIXED
TKTN   03.3   SYSPAR 248. 00010000 LB0:-01126742
RMDEMO X03.03 GEN    225. 00013700 SY0:-00352100
MTAACP 0006   GEN    200. 00013000 SY0:-00000777
F11MSG V0010  GEN    200. 00005400 SY0:-01053030
MCR... 3.2    SYSPAR 160. 00010000 LB0:-01051626
DCL... 0113A  SYSPAR 160. 00010000 LB0:-00567017 CHECKPOINTED
...DCL 0113A  GEN    160. 00040000 LB0:-01221102
TAST17 01     GEN    160. 00025000 LB0:-01051676
...STO 000005 SYSPAR 151. 00010000 SY0:-01136540
...BYE 01.5   GEN    150. 00040000 LB0:-01051337
F11ACP M0235  FCPPAR 149. 00040000 LB0:-01052445
...DMO  3.2   GEN    140. 00040000 LB0:-01051403
...INI 03.1   GEN    140. 00040000 LB0:-01051561
...UFD V0407  GEN    140. 00040000 LB0:-0105
<CTRL/O>   prekinjen izpis
```

5.2.6 LISTA INSTALIRANIH TASKOV-AL FORMAT

Na zaslon nam izpiše imena in status vseh instaliranih taskov v ATL formatu. (Glej opis ukaza ATL za pojasnitev izpisa)

Oblika ukaza:

TAL [ime-taska]

Kjer je ime-taska ime taska, ki nas zanima. Če ime taska spustimo, nam izpiše za vse taske.

Primer:

.2.7 OPIS PARTICIJ - PAR

kaz izpiše na terminal razdelitev pomnilnika na particije. Izpis je oblikovan v pet stolpcev, ki pomenijo:

1. ime particije
2. bazno adreso particije (osmiško)
3. velikost pasticije v bytih (osmiško)
4. vrsta particije :glavna (MAIN), podparticija (SUB)
5. tip particije:

TASK	uporabniško kontrolirana
COM	skupno področje
DEV	za registre enot
SYS	sistemsko kontrolirana
ime taska	področje taska
DYNAMIC	dinamično kreirana področja
DRIVER	Področje zasedeno z driverjem

OM particije so za rezidentne biblioteke in skupna podatkovna oročja. DEV particije služijo taskom za komunikacijo s posebni- i enotami preko registrov.

blika ukaza:

```
PARtitions
```

rimer:

```
IME     BAZA   DOLŽINA VRSTA TIP
---     ----   ------- ----- ---
>PAR
LDRPAR  100000 002500  MAIN  TASK
TTPAR   102500 020000  MAIN  TASK
DRVPAR  122500 015300  MAIN  SYS
        122500 001200  SUB   DRIVER -DK:
        123700 002100  SUB   DRIVER -DL:
        126000 002400  SUB   DRIVER -DY:
        130400 001100  SUB   DRIVER -LP:
        131500 003300  SUB   DRIVER -MT:
        135000 003000  SUB   DRIVER -CO:
SYSPAR  140000 010000  MAIN  TASK
FCPPAR  150000 024100  MAIN  SYS
        150000 024100  SUB   (F11ACP)
SHFPAR  174100 002000  MAIN  TASK
GEN     176100 561700  MAIN  SYS
        176100 100000  SUB   (...BAS)
        276100 020000  SUB   (...MCR)
        435600 020400  SUB   (QMG...)
        456200 014600  SUB   (COT...)
```

V A J E

1. Razvij program FORTRAN.FTN in ga izvedi.

2. Razvij program PASKAL.PAS in ga izvedi.

3. Izvedi program FORTRAN tako, da se bo izvajal po imenom PRIMER in ne pod imenom tvojega terminala.

4. Preveri, kateri taski so aktivni v računalniku.

5. Startaj program FORTRAN in ga nasilno prekini z ukazom ABO. Startaj ga na isti način kot v vaji 3 in ga spet nasilno prekini.

6. Izpiši imena vseh instaliranih taskov na ekran.

7. Izpiši imena vseh aktivnih taskov v sistemov tako, da boš lahko odčital tudi statuse in vrednosti registrov.

8. Izpiši imena particij sistema.

POGLAVJE6

POSREDNE UKAZOVNE DATOTEKE

V tem poglavju se bomo seznanili s posrednimi ukazovnimi datotekami in načinom njihove uporabe.Posredna ukazovna datoteka je tekstovna datoteka, ki vsebuje niz ukazov namenjenih določenemu tasku, ki jih interpretira. Ta task je navadno eden izmed sistemskih taskov operacijskega sistema DELTA/M, kot so: MCR, MACRO-11 ali TKB.

Imamo dve vrsti ukazovnih datotek: task posredne ukazovne datoteke in MCR posredne ukazovne datoteke. Seznanili se bomo le z MCR posrednimi ukazovnimi datotekami. Posredne MCR ukazovne datoteke vsebujejo niz MCR ukazov. Poleg teh lahko taka datoteka vsebuje še posebne ukaze za kontrolo izvajanja posrednih ukazovnih datotek.

Izvajanje posrednih ukazovnih datotek sprožimo tako, da pred ime datoteke postavimo znak @. Primer:

```
>@PRIMER.CMD
```

Privzeta vrednost za tip datoteke pri posrednih ukazovnih datotekah je .CMD. Tako lahko navedeni primer zapišemo:

```
>@PRIMER
```

Ko se konča izvajanje ukazovne datoteke, izpiše :

```
>@<EOF>
```

Če želimo prekiniti izvajanje posredne ukazovne datoteke pritisnemo CTRL/Z, ko zahteva kak vnos. Spet izpiše sporočilo:

```
>@<EOF>
```

6.1 MCR PROCESOR POSREDNIH UKAZOVNIH DATOTEK

MCR ima poseben procesor (AT.) za interpretiranje ukazov v ukazovnih datotekah. Posredna MCR ukazovna datoteka lahko poleg MCR ukazov vsebuje posebne ukaze (takoimenovane direktive), ki jih ta procesor razume in interpretira.

Ko izvajamo posredno ukazovno datoteko, AT. najprej prečita datoteko in interpretira vrsto, kot bi bila podana neposredno MCR-u ali kot zahteva samemu AT. za kak poseg. Direktive se razlikujejo od ostalih vrstic po tem, da imajo piko (.) kot prvi znak v vrstici. MCR ukazi nimajo nobenega posebnega znaka pred sabo.

AT. direktive tvorijo poseben proceduralni jezik, ki nam omogoča:

1. definiranje label
2. definiranje in prirejanje vrednosti simbolom treh tipov: logičnim, numeričnim in nizovnim
3. kreiranje in pristopanje podatkovnim datotekam
4. kontrolo izvajanja
5. logično testiranje
6. vspostavljanje in ukinjanje vsakega od šestih načinov dela
7. povečevanje in zmanjševanje vrednosti numeričnih simbolov
8. kontrolo pararelno izvajanih taskov ter taskov,ki se izvajajo na časovni osnovi

Če definiramo nek simbol, AT. procesor naredi vpis v tabelo simbolov. Ta vrednost se nahaja v tabeli simbolov ves čas izvajanja ukazovne datoteke.

MCR izpiše vsak ukaz na terminal preden ga izvede. Če pa sprožimo AT. z ime-datoteke/-MCR, se MCR komande samo izpišejo kot komentar s (!) klicajem spredaj in se ne izvedejo.

MCR posredna ukazovna datoteka lahko vsebuje tudi komentar, ki ga procesor izpisuje na terminal. Komentarji, ki začenjajo vrstico se morajo začeti z podpičjem (;) (pred podpičjem je še lahko direktiva). Če želimo postaviti komentar za nekim MCR ukazom, moramo pred komentar postavoti klicaj (!). Vse vrstice, ki se začnejo s piko in ki ji sledi podpičje, AT. procesor razume kot komentar, ki ga ne izpisuje, ko izvaja posredno ukazovno datoteko.

Posredni ukazovni procesor veže terminal nase, ko izpisuje komentar, ki se začenja s podpičjem. Če je več vrstic komentarja lahko s CTRL/O preprečimo njegov izpis. Izpis se nadaljuje, če ponovno pritisnemo CTRL/O ali če AT. naleti na MCR ukaz.

6.2 SIMBOLI

AT. procesor nam dovoljuje uporabo simbolov. Te simbole lahko potem testiramo ali pa primerjamo med sabo v smislu nadaljnjega izvajanja posredne ukazovne datoteke. Z numeričnimi simboli lahko tudi računamo med tem ko nizovne simbole lahko sestavljamo. Lahko jih uporabljamo in zamenjujemo v MCR ukazih, kot zapise v podatkovnih datotekah in komentarjih.

Imena simbolov so ASCII nizi do 6 znakov dolžine. Začeti se morajo z črkami A-Z ali $; vsebujejo lahko alfanumerične znake in $.

Imamo tri tipe simbolov:

* logični
* nizovni
* numerični

Logični simbol lahko ima vrednost ali pravilno (TRUE) ali narobe (FALSE).

Nizovni simboli lahko imajo od 0 do 80 ASCII znakov.

Numerični simbol lahko ima vrednost od 0 do 65535. Lahko imajo osmiško ali desetiško vrednost. Ta razlika je pomembna samo, če spreminjamo vrednost simbola.

Tip simbola (logični, nizovni, numerični) se določi, ko mu prvič priredimo neko vrednost z eno od prireditvenih direktiv. Vse nadaljnje prireditve mu lahko priredijo različne vrednosti ne pa tudi tip. Prireditvene direktive lahko priredijo:

* TRUE ali FALSE vrednost k logičnemu simbolu (.ASK, .SETT, .SETF)
* osmiško ali desetiško število numeričnemu simbolu (.ASKN, .SETN)
* znakovni niz nizovnemu simbolu (.ASKS, .SETS)

6.2.1 POSEBNI SIMBOLI

AT. procesor avtomatično definira posamezne posebne simbole odvisno od stanja sistema in dogodkov med izvajanjem posredne ukazovne datoteke. Posebne simbole lahko testiramo, primerjamo ali pa spreminjamo njihove vrednosti. Obstajajo trije tipi posebnih simbolov: logični, numerični in nizovni. Vsi posebni simboli imajo enako obliko. Ime simbola je med znakom < in znakom >. Tukaj jih bomo spoznali le nekaj:

<EXSTAT>

je posebni numerični simbol in dobi vrednost 0,1,2 ali 4 odvisno od tega, kako se je končalo izvajanje zadnjega MCR ukaza ali .WAIT direktive, ki se je nanašala na task, ki smo ga poslali na izvajanje z direktivo .XQT.

Simbol <EXSTAT> spreminja vrednost sinhrono z izvedbo MCR ukaza ali .WAIT direktive. Vrednost simbola lahko spremeni tudi direktiva .EXIT. Simbol dobi vrednost, če se je izvajanje taska končalo s statusom. V ostalih primerih dobi simbol vrednost od MCR-a. Vrednosti 0,1,2 in 4 pomenijo:

0 - opozorilo
1 - uspešno
2 - napaka
4 - resna napaka

<STRLEN>

dobi vrednost dolžine niza, ki smo ga nazadnje vnesli preko .ASKS direktive

<DATE>

je posebni nizovni simbol, ki vsebuje tekoči datum; format dd-mes-ll

<TIME>

je posebni nizovni simbol, ki priredi tekoči čas v obliki hh:mm:ss

6.2.2 VSTAVLJANJE VREDNOSTI SIMBOLOV

V posrednih ukazovniih lahko uporabljamo vrednosti simbolov (logičnih, nizovnih, numeričnih in posebnih) kot del ukaza ali kot parameter ukaza tako, da parameter ali del ukaza nadomestimo z imenom simbola v enojnih narekovajih. Da bi AT. procesor lahko izvedel zamenjavo, moramo še prej izdati direktivo .ENABLE SUBSTITUTION. AT. procesor zamenja ime simbola med narekovaji z vrednostjo simbola.

Ko AT. odkrije narekovaj (‘), vzame ves tekst do sledečega narekovaja kot ime simbola. Potem poišče v tabeli simbolov vrednost tega simbola ter jo v ukazovni vrstici zamenja za ime simbola skupaj z narekovaji.

Primer:imejmo sledečo ukazovno datoteko

```
.ASKS DEV MOUNTAJ NA ENOTO?
.ENABLE SUBSTITUTION
MOUNT 'DEV'
```

Ko AT. izvaja to ukazovno datoteko, izpiše:

```
>* MOUNTAJ NA ENOTO? [S]: DK2:
>MOUNT DK2:
```

DK2: smo vnesli kot odgovor na vprašanje. Niz DK2: smo priredili simbolu DEV. Ko je AT. prečital MOUT 'DEV', je 'DEV' zamenjal z vrednostjo simbola DEV, to je DK2:. Če substituiranje ne bi bilo dovoljeno (z .ENABLE SUBSTITUTION), bi AT. poskušal izvesti MOUNT 'DEV', kjer bi za ime enote vzel 'DEV'.

Ker pa je bilo substituiranje dovoljeno, je je tekst med narekovajema razumel kot ime simbola. Če bi želeli v nek tekst vključiti tudi apostrof, moramo enojni apostrof nadomestiti z dvema zaporednima apostrofoma. Primer:

```
! DA B'' UKA ŽEJA ME IZ TVOJGA SVETA
```

izpiše kot

```
! DA B' UKA ŽEJA ME IZ TVOJGA SVETA
```

6.2.3 NUMERIČNI IN NIZOVNI SIMBOLI

AT. procesor lahko prireja simbolom ali numerične ali nizovne vrednosti. S prirejanje določimo tudi tip simbola, ki ga ne moremo menjati.

6.2.3.1 Numerični simboli in izrazi -

Numerični simbol je niz števil, ki predstavljajo vrednost od 0 do 177777 osmiško (oz. od 0 do 65535 desetiško, če za številom stoji pika). Če s kako aritmetično operacijo prestopimo to mejo, AT. javi usodno napako in izpiše sporočilo:

```
AT.--NUMERIC UNDER - OVERFLOW
```

Ko numerični simbol spravimo v niz, se tip (desetiško ali osmiško) določi na osnovi uporabljene direktive .ASKN ali .SETN .

Numerične simbole in konstante lahko s pomočjo aritmetičnih in logičnih operatorjev povezujemo v numerične izraze. Uporabljati smemo aritmetične operatorje za seštevanje (+), odštevanje (-), množenje (*) in deljenje (/).

Numerični izrazi se izvajajo z leve na desno; lahko pa to spremenimo z oklepaji. Primer: direktive

```
.SETN N1 2
.SETN N2 3
```

```
.SETN N3 N1+N2*4
```

priredijo simbolu N3 vrednost 24 osmiško medtem ko direktive

```
.SETN N1 2
.SETN N2 3
.SETN N3 N1+(N2*4)
```

priredijo simbolu N3 vrednost 16 osmiško.

Tip izraza je osmiški, če so vsi operandi osmiški; sicer je tip desetiški. Direktive

```
.SETN N1 2
.SETN N2 3
.SETN N3 N1+N2*4.
```

priredijo simbolu N3 vrednost 20 desetiško.

Numerični izrazi so dovoljeni kot drugi operand v .IF in .SETN direktivah. Pravtako so taki izrazi dovoljeni kot argumenti za določanje intervala ali defaulta v .ASKS in .ASKN direktivah. .EXIT in .STOP direktivi lahko imata za specifikacijo izhodnega statusa pravtako numerični izraz.

6.2.3.2 Nizovni simboli, podnizi in izrazi -

Nizovna konstanta je vsaka skupina izpisljivih znakov med dvema apostrofoma ('). Dolžina niza ne sme biti večja od 80, lahko pa je niz prazan.Primer:

```
'ABCDEF'
''
```

Nizovni simboli lahko imajo vrednost katere koli nizovne konstante. Vrednost jim prirejamo z direktivama .ASKS ali .SETS. Direktivi

```
.SETS S1 'ABCDEF'
.SETS S2 S1
```

priredita simbolu S2 vrednost simbola S1 to je 'ABCDEF'

Imamo tudi možnost izločanja podnizov iz nizov. Podnize lahko uporabljamo le kot drugi operand v .IF in .SETS direktivi. Primer:Direktivi

```
.SETS S1 'ABCDEF'
.SETS S2 S1[1:3]
```

priredita simbolu S2 vrednost simbola S1 od 1 do vključno tretjega znaka, to je 'ABC'.

Kombiniramo pa lahko tudi nize in podnize ter nizovne simbole drug z drugim s pomočjo znaka za povezovanje (+) v nizovne izraze.

Nizovni izrazi so dovoljeni kot drugi operand v direktivah .SETS in .IF, kjer pa mora biti prvi operand nizovni simbol. Primer: direktive

```
.SETS S1 "A"
.SETS S2 "CDEF"
.SETS S3 S1+"B"+S2[1:3]
```

priredi simbolu S3 vsebino simbola S1, konstante "B" in prve tri znake simbola S2 (to je "ABCDE")

6.3 DIREKTIVE

Direktive morajo biti ločene od svojih argumentov in MCR ukazov za vsaj en prazen znak. Na tri mesta v ukazovni vrstici lahko vnesemo poljubno število praznih znakov ali TAB-bov:

* na začetek ukazovne vrstice

* neposredno za dvopičjem (:) pri labeli

* na koncu ukazovne vrstice

To nam omogoča večjo preglednost ukazovne datoteke. Pametno je pisati labele v prvo kolono ostalo pa v deveto (uporabimo en TAB).

Pomembna izjema so vrstice med .ENABLE in .DISABLE DATA direktivama. Vsak prazen znak ali TAB postane del podatkov v datoteki, ki smo jo odprli.

6.3.1 RAZDELITEV DIREKTIV

Direktive lahko razdelimo v več skupin z obzirom na njihovo delovanje.

Definicija label

.labela:

povežе vrstico z nizom, ki smo ga navedli kot labelo, tako, da moremo preko imena pristopati k tej vrstici

Definiranje smbolov

.ASK

definira ali redefinira logični simbol; damo mu logično vrednost (TRUE ali FALSE)

.SETT/.SETF

definira ali redefinira logični simbol; priredi mu TRUE/FALSE vrednost

.ASKN

definira ali redefinira numerični simbol; damo mu neko spremenljivo numerično vrednost

.SETN

definira ali redefinira numerični simbol; priredi mu numerično vrednost

.DEC

zmanjša vrednost numeričnega simbola za 1

.INC

poveča vrednost numeričnega simbola za 1

.ASKS

definira ali redefinira nizovni simbol; damo mu zaporedje ASCII znakov

.SETS

definira ali redefinira nizovni simbol; priredi mu niz ASCII znakov

Logična kontrola:

.GOTO

izvajanje se nadaljuje na navedeni labeli

.GOSUB

preskoči na podproceduro v isti ukazovni datoteki, ki se začne z navedeno labelo

.RETURN

pomeni konec neke podprocedure; izvajanje se nadaljuje z vrstico, ki se nahaja za .GOSUB direktivo, s katero smo preskočili na to podproceduro

.EXIT

prekine izvajanje posredne ukazovne datoteke in eventuelno še priredi vrednost simbolu <EXSTAT>

.STOP

prekine se izvajanje ukazovne datoteke in v nekaterih primerih AT. priredi izhodni status

Logični testi

.IF

ugotovi ali pogoj ustreza enemu ali večim pogojem

.IFT/.IFF

ugotovi ali je vrednost logičnega simbola TRUE ali FALSE

Vspostavljanje/ukinjanje načinov delovanja

.DISABLE

ukinemo eno od sledečih možnosti: substitucija simbolov, uporabo .DATA direktive, globalne simbole, pretvarjanje malih črk v velike, prepoznavanje ESC, odzivnost na zaslonu

.ENABLE

vspostvimo eno od sledečih možnosti: substipucija simbolov, uporabo .DATA direktive, globalne simbole, pretvarjanje malih črk v velike, prepoznavanje ESC, odzivnost na zaslonu

Pristop k datotekam

.OPEN

odpre in tvori izhodno podatkovno datoteko (če obstaja, tvori novo verzijo)

.OPENA

odpre obstoječo podatkovno datoteko in doda podatke, ki sledijo

.DATA

specificira eno vrstico podatkov za prenos na izhodno datoteko

.CLOSE

zapre izhodno podatkovno datoteko

Kontrola izvajanja

.DELAY

prekine izvajanje posredne ukazovne datoteke za navedeni časovni interval

.PAUSE

prekine izvajanje posredne ukazovne datoteke in omogoča uporabniku, da naredi nek poseg

.XQT

sproži izvajanje navedenega taska in nadaljuje izvajanje posredne ukazovne datoteke

.WAIT

čaka, da navedeni task konča z izvajanjem

6.3.2 DEFINIRANJE LABEL

Labelo vedno postavljamo na začetek vrstice; v isti vrstici je še lahko direktiva in/ali MCR ukaz ali komentar. Ko se kontrola prenese na neko labelo, se izvede, karkoli je takoj za podpičjem. Ukaza ni potrebno ločevati od labele z praznim znakom. V vrstici je dovoljena samo ena labela. Je lahko dolga 1 do 6 alfanumeričnih in $ znakov.

Labelo lahko definiramo tudi kot direktno pristopno labelo. Ko enkrat procesor uporabi tako labelo, si jo zapomni; naslednji pristop k taki labeli je mnogo hitrejši. Če je labela direktno pristopna, se izvede, ko na tako labelo preskočimo, vrstica takoj za to labelo. Definiramo lahko le do 20(10) direktno pristopnih label. Vsaka nadaljna prekrije najzgodnejšo med obstoječimi. Da bi neka labela bila direktno pristopna, mora biti vrstica za dvopičjem prazna. Primer:

```
.100:   .ASK A ALI ŽELIŠ NADALJEVATI?
.IFT    A       .GOTO 200
          .
          .
          .
.200:
;TO JE ZAČETEK POD RUTINE
          .
          .
          .
.RETURN
```

.3.3 DEFINIRANJE IN REDEFINIRANJE VREDNOSTI SIMBOLOV

I direktivami lahko definiramo ali pa spremenimo vrednosti simbolom. Ko prvič uporabimo neko ime simbola, se to ime vpiše v tabelo simbolov, priredi se mu vrednost in določi tip, ki sa pozneje e smemo spreminjati.

.3.3.1 Definiranje in redefiniranje vrednosti logičnega simbola -

Logični simbol lahko definiramo ali spremenimo njegovo vrednost z direktivami .ASK, .SETT in .SETF.

.3.3.1.1 Direktiva .ASK -

Direktiva .ASK izpiše vprašanje na terminal, počaka odgovor in setira simbol kot TRUE ali FALSE odvisno od odgovora. Če simbol še ni bil definiran, se vpiše v tabelo simbolov. Če je tak logični simbol že obstajal, se setira odgovoru ustrezna vrednost. Če je bil simbol prej definiran kot nizovni ali numerični, procesor javi usodno napako.

Format:

```
.ASK ssssss tekst
```

kjer je:

ssssss 1 do 6 znakov za ime simbola, h kateremu bo prirejena vrednost TRUE/FALSE

teskt kakršno koli zaporedje ASCII znakov z praznim znakom spredaj; največ 70

Ko se izvede .ASK direktiva, procesor izpiše tekstovni-niz z zvezdico spredaj in dodatkom '? [D/N]:' na koncu. Procesor prepoznava naslednje odgovore:

1. D<CR> - setira simbol ssssss kot TRUE
2. N<CR> - setira simbol ssssss kot FALSE
3. <CR> - setira simbol ssssss kot FALSE
4. <ESC> - setira simbol ssssss kot TRUE

Primer: direktiva

```
.ASK INSPIP ALI ŽELIŠ INSTALIRATI PIP
```

izpiše

>* ALI ŽELIŠ INSTALIRATI PIP? [D/N]:

Simbol INSPIP bo setiran TRUE ali FALSE odvisno od odgovora, ki ga bomo dali na vprašanje.

6.3.3.1.2 Dirktivi .SETT in .SETF -

Direktivi .SETT/.SETF definirata ali redefinirata vrednost logičnega simbola. Če simbol ni bil definiran, procesor vpiše simbol v tabelo simbolov in mu da ustrezno vrednost. Če je bil simbol že definiran, mu direktiva priredi usrezno vrednost. Procesor javi napako, če poskušamo redefinirati numerični ali nizovni simbol.

Oblika ukaza:

.SETT ssssss

.SETF ssssss

kjer je ssssss 1 do 6 znakovno ime simbola; pred in za imenom mora biti vsaj en prazen znak.

Primeri:

.SETT X

setira logični simbol X kot TRUE.

.SETF ABCD

setira logični simbol ABCD kot FALLSE.

6.3.3.2 Definiranje in redefiniranje numeričnih simbolov -

Vrednost numeričnega simbola lahko določimo z direktivama .ASKN in .SETN.

6.3.3.2.1 Direktiva .ASKN -

Direktiva .ASKN izpiše na terminal zahtevo za vnos numerične vrednosti, počaka, da jo vnesemo, preveri ali je na določenem intervalu ali pa priredi privzeto vrednost simbolu, ki smo ga navedli. Če simbol prej ni obstajal, ga vpiše v tabelo simbolov. Če pa je že obstajal, mu priredi odgoru ustrezno vrednost. AT. Javi usodno napako, če je bil simbol prej definiran kot logičen ali nizovni.

Format:

.ASKN ssssss tekst

.ASKN [sm:zm] ssssss tekst

.ASKN [sm:zm:pv] ssssss tekst

.ASKN [::pv] ssssss tekst

(Oklepaji so del sintakse)

kjer je:

ssssss od 1 do 6 znakovno ime simbola, ki mu prirejamo vrednost

tekst zaporedje ASCII znakov (do 70) z vsaj enim praznim znakom spredaj, ki se izpiše ob izvajanju direktive.

sm:zm spodnja (sm) in zgornja (zm) meja dovoljenega odgovora (vključno)

pv privzeta vrednost

kot stm, zm in pv so lahko kombinacije konstant in numeričnih simbolov z drugimi konstantami in numeričnimi simboli - aritmetični izrazi.

Ukazovna vrstica ne more biti daljša od 80 znakov. Ko se izvede .ASKN direktiva, se tekstovni-niz izpiše z zvezdico spredaj ter [OI: zadaj, kar pove, da mora biti odgovor osmiški ali [DI: zadaj, kar pove, da mora biti odgovor desetiški. Odgovor mora biti v navedenem intervalu ali v intervalu 0-177777 osmiško oz. 0-65535 desetiško.

Če je odgovor izven tega območja, procesor javi:

AT. --VALUE NOT IN RANGE

in ponovi zahtevo za vnos simbola.

Če v primeru, ko sm, zm ali pv izračunavamo, dobimo vrednost,ki je večja od 177777(8), procesor javi

AT. --NUMERIC UNDER- OR OVER-FLOW

Če je odgovor prazna vrstica in privzeta vrednost ni predpisana, se privzame 0. V tem primeru mora interval vsebovati tudi 0.

Odgovor je lahko osmiški ali desetiški; znak pred številom pomeni, da je odgovor osmiški, pika (.) zadaj pa, da je desetiški. Če oboje izpustimo, se privzame desetiško, če so meje ali privzeta vrednost določene desetiško (s piko na koncu). V ostalih primerih se smatra vnos osmiški.

če želimo doseči desetiški vnos ne da bi določili interval ali privzeto vrednost, zapišemo:

```
.ASKN [:0.] VNESI VREDNOST
```

Ko vnesemo vrednost za simbol, AT. določi tip (osmiški ali desetiški) in sa vse k simbolu, vrednost pa zapiše kot niz.

Primeri:

direktiva

```
.ASKN SYM DEFINIRAJ NUMERIČNI SIMBOL
```

izpiše na terminal

```
>*DEFINIRAJ NUMERIČNI SIMBOL [O]:
```

[O] = privzeti tip (osmiški)

Procesor definira simbol SYM in mu dodeli vnešeno vrednost.

Direktiva

```
.ASKN [2:35:16] NUMSYM DEFINIRAJ NUMERIČNI SIMBOL
```

izpiše tekstovni-niz

```
>*DEFINIRAJ NUMERIČNI SIMBOL [O R:2-35 D:16]:
```

z formatom [x R:sm-zm D:pv], kjer je

x	je O če je privzeti tip osmiški, ali D če je privzeti tip desetiški
R:sm-zm	označuje interval
D:pv	označuje privzeto vrednost

Procesor preveri ali je vnešena vrednost v predpisanem intervalu.

Direktiva

```
.ASKN [NSYM+10:45:NSYM+10] SYM DEFINIRAJ NUMERIČNI SIMBOL
```

izpiše (če privzamemo za NUMSYM vrednost 16)

```
>*DEFINIRAJ NUMERIČNI SIMBOL [O R:26-45 D:26]:
```

6.3.3.2.2 Direktiva .SETN

Direktiva .SETN definira ali redefinira vrednost numeričnega simbola. Če simbol ni bil definiran, procesor vpiše simbol v tabelo simbolov in mu da ustrezno vrednost. Če je bil simbol že definiran, mu direktiva priredi usrezno vrednost. Procesor javi napako, če poskušamo redefinirati losični ali nizovni simbol.

Oblika ukaza:

.SETN ssssss numerišni-izraz

kjer je ssssss 1 do 6 znakovno ime simbola; pred in za imenom mora biti vsaj en prazen znak.

Vrednost numeričnega izraza se priredi simbolu. Izraz je lahko sestavljen iz konstant ali simbolov, ki jih povezujejo numerični operatorji. Operacije se izvajajo z leve proti desni; lahko uporabljamo oklepaje. Tip izraza ji osmiški, če so vsi operandi osmiški. Če je le eden od operandov desetiški, je tip izraza desetiški.

Primeri:

.SETN X 27

priredi numeričnemu simbolu X kot vrednost 27 (osmiško)

.SETN X1 3*(X2-5)

Direktiva priredi simbolu X1 vrednost (X2-5)*3 .

6.3.3.2.3 Direktiva .INC

Direktiva .INC poveča numerični simbol za 1. Če je bil simbol definiran kot losični ali nizovni, procesor javi usodno napako.

Oblika ukaza:

.INC ssssss

kjer je ssssss 1 do 6 znakovno ime numeričnega simbola.

.INC N

Vrednost simbola N se poveča za 1.

6.3.3.2.4 Direktiva .DEC -

Direktiva .DEC zmanjša numerični simbol za 1. AT. javi usodno napako, če je bil simbol definiran kot losičen ali nizovni. Oblika ukaza: .DEC ssssss kjer je ssssss ime sibola (1 do 6 znakov) Primer: .DEC N Ta direktiva zmanjša simbol N za 1.

6.3.3.3 Definiranje in redefiniranje nizovnega simbola -

Vrednost nizovnega simbola lahko definiramo ali pa spremenimo z direktivama .ASKS in .SETS.

6.3.3.3.1 Direktiva .ASKS - Direktiva .ASKS izpiše na terminal zahtevo za vnos nizovnega simbola in počaka, da ga vnesemo. Če je potrebno, preveri ali je dolžina na določenem intervalu. Če smibol prej ni obstajal,ga vpiše v tabelo simbolov. Če pa je že obstajal, mu priredi odgovoru ustrezno vrednost. AT. javi usodno napako, če je bil sim- bol prej definiran kot losičen ali numeričen. Če je dolžina vnešenega niza izven navedenega intervala, izpiše sporočilo:

AT. --STRING LENGTH NOT IN RANGE

ter ponovno izpiše zahtevo za vnos.

Format:

.ASKS ssssss tekst

.ASKS [sm:zm] ssssss tekst

(Oklepaji so del sintakse)

kjer je:

ssssss od 1 do 6 znakovno ime simbola, ki mu prirejamo vrednost

tekst zaporedje ASCII znakov (do 70) z vsaj enim praznim znakom spredaj, ki se izpiše ob izvajanju direktive.

sm:zm meje dovoljene dolžine niza, ki ga vnašamo(vključno)

sm in zm sta lahko kombinaciji konstant in numeričnih simbolov z drugimi konstantami in numeričnimi simboli - aritmetični izrazi .

Ukazna vrstica ne more biti daljša od 80 znakov. Ko se izvede .ASKS direktiva, se tekstovni-niz izpiše z zvezdico spredaj ter [S]: zadaj, kar pove, da mora biti odgovor ASCII niz znakov.

Primeri:

direktiva

.ASKS IME PROSIM, VNESI SVOJE IME

izpiše na terminal

>*PROSIM, VNESI SVOJE IME [S]:

Procesor definira simbol IME in mu dodeli vnešeni niz.

Direktiva

.ASKS [1:15] PRII PROSIM, VNESI SVOJ PRIIMEK

izpiše tekstovni-niz v obliki [S R:sm-zm]

>*PROSIM, VNESI SVOJ PRIIMEK [S R:1-15]:

S tip simbola (string)

R:sm-zm določa interval za število znakov

6.3.3.3.2 Direktiva .SETS

Direktiva .SETS definira ali redefinira vrednost nizovnega simbola. Če simbol ni bil definiran, procesor vpiše simbol v tabelo simbolov in mu da ustrezno vrednost. Če je bil simbol že definiran, mu direktiva priredi usrezno vrednost. Procesor javi napako, če poskušamo redefinirati numerični ali logični simbol.

Oblika ukaza:

.SETS ssssss nizovni-izraz

kjer je ssssss 1 do 6 znakovno ime simbola; pred in za imenom mora biti vsaj en prazen znak.

Procesor priredi simbolu vrednost nizovnega izraza. Če v izrazu nastopajo tudi konstante, morajo biti med narekovaji ('). V izrazih lahko kombiniramo nizovne simbole, podnize in konstante z operatorjem za povezovanje (+).

Primeri:

.SETS A 'ABCDEF'

priredi nizovnemu simbolu simbolu A niz 'ABCDEF'

.SETS A1 'ZZZZ'

priredi nizovnemu simbolu simbolu A1 niz 'ZZZZ'

```
.SETS A2 A1+'ABCD'
```

Priredi nizovnemu simbolu simbolu A2 niz sestavljen iz niza A1 in 'ABCD'.

```
.SETS A2 A1+A[1:4]
```

Ta direktiva naredi isto kot prejšnja: nizu A2 priredi niz A1 in doda prve štiri znake niza A.

6.3.4 DIREKTIVE ZA LOGIČNO KONTROLO

Z direktivami za logično kontrolo izvajamo vejanje in prekinjamo izvajanje posredne ukazovne datoteke.

6.3.4.1 Direktive za vejanje -

6.3.4.1.1 Direktiva .GOTO -

Direktiva .GOTO povzroči preskok na navedeno labelo. Na ukaze med direktivo in labelo, na katero skače, se ne ozira. Skok se lahko izvrši naprej ali nazaj.

Oblika ukaza:

```
.GOTO labela
```

kjer je 'labela' ime labele brez pike spredaj in dvopičja zadaj. Pred labelo mora biti vsaj en prazen znak.

Primer:

```
.GOTO 100
```

prenese kontrolo na labelo .100:.

6.3.4.1.2 Klicanje podprocedur- direktiva .GOSUB -

Direktiva .GOSUB zapiše trenutno mesto v ukazovni datoteki in preskoči na navedeno labelo. Labela označuje začetek podprocedure, ki se konča z .RETURN direktivo. AT si zapomni stanje in nato poišče prvo pojavljanje labele naprej po datoteki. Direktiva .RETURN vrne kontrolo v vrstico, ki sledi je osem.

oblika ukaza:

.GOSUB labela

kjer je labela prva vrstica podprocedure. V direktivi .GOSUB labelo pišemo brez pike spredaj in dvopičja zadaj, od direktive pa jo mora ločevati vsaj en prazen znak.

Primer:

.GOSUB LAB

Navedena direktiva prenese kontrolo na labelo .LAB:

6.3.4.1.3 Direktiva .RETURN -

Direktiva .RETURN pomeni konec podprocedure; kontrola se prenese na vrstico iza .GOSUB direktive, s katero smo preskočili na to podproceduro.

Oblika ukaza:

.RETURN

6.3.4.2 Direktive za prekinitev izvajanja -

6.3.4.2.1 Direktiva .EXIT -

Direktiva .EXIT prekine izvajanje tekoče ukazovne datoteke in kontrolo prevzame MCR. Imamo tudi možnost, da damo simbolu <EXSTAT> neko vrednost.

Oblika ukaza:

.EXIT [vrednost]
(oklepaji niso del sintakse)

kjer je vrednos, ki jo bo dobil <EXSTAT> simbol.

.EXIT

če ima direkriva še številski izraz

.EXIT N+2

AT. prenese vrednost,ki jo izračuna v simbol <EXSTAT>.

6.3.4.2.2 Direktiva .STOP -

Direktiva .STOP prekine izvajanje posredne ukazovne datoteke in kontrolo prevzame MCR. Izpiše se sporočilo

>●<EOF>

Dierktiva nam tudi omogoča postavljanje izhodnega statusa procesorja AT.

Oblika ukaza:

.STOP [vrednost]

Vrednost pomeni izhodno izhodni status procesorja. Če takega statusa ne specificiramo, ima .STOP direktiva enak učinek kot direktiva logičnega konca (/).

Primer:

.STOP 0

Direktiva prekine izvajanje posredne datoteke in izhodni status procesorja AT. je 0.

6.3.4.2.3 Direktiva za logični konec -

Direktiva za logičen konec (/) povzroči, da se konča izvajanje posredne ukazovne datoteke, in procesor izpiše

>●<EOF>

Oblika ukaza:

/

mora biti prvi neprazen znak v vrstici. To direktivo lahko uporabljamo kjerkoli v ukazovni datoteki, vendar zna biti konec zelo nepredvidljiv.

Primer:

```
        .ASK A ZELIS NADALJEVATI
        .IFT A .GOTO 100
        /
.100:
```

6.3.5 LOGIČNI TESTI

AT. procesor pozna več direktiv za logične teste. Testiramo lahko logične simbole, prisotnost taskov v sistemu itd. Če je rezultat testa pozitiven, se izvede ostanek vrstice. Če je rezultat negativen, se izvede sledeča vrstica. Enostavne logične teste lahko združujemo z direktivama .AND in .OR.

6.3.5.1 Testiranje vrednosti simbolov - direktiva .IF -

Direktiva .IF primerja numerični oz. nizovni simbol z izrazom istega tipa in ugotovi ali je pogoju zadovoljeno; če je, procesor izvede preostanek vrstice; če ni, gre v sledečo vrstico. Če primerja nizovni simbol z nizovnim izrazom, primerja ASCII vrednosti znakov operandov z leve proti desni. En od operandov se smatra večji, če ima prvi znak večjo ASCII vrednost kot enakoležeči v drugem operandu. Numerične simbole primerja po velikosti.

Oblika ukaza:

```
.IF simbol relop izraz {MCR ukaz }
                       {direktiva}
                       {komentar }
```

kjer pomeni:

simbol 1 do 6 znakovno ime numeričnega ali nizovnega simbola

relop eden od sledečih relacijskih operatorjev

EQ ali =	-enako
NE ali <>	-neenako
GE ali >=	-večje ali enako
LE ali <=	-manjše ali enako
GT ali >	-večje
LT ali <	-manjše

izraz izraz, sibmol ali konstanta enakega tipa kot simbol

{MCR ukaz / direktiva / komentar} MCR ukaz, direktiva ali komentar, ki se izvede oz.izpiše, če je pogoj izpolnjen

Primeri:

```
.SETS X "A"
.SETS Y "a"
.IF X LT Y .GOTO 200
```

ASCII vrednost nizovnega simbola X je manjša od ASCII vrednosti nizovnega simbola Y, kar zadovoljuje pogoj in zato se izvede .GOTO 200.

```
.SETN N1 2
.SETN N2 7
.IF N1 <= N2 PIP/LI
```

Ker je pogoj izpolnjen, se izvede ukaz PIP.

```
.SETS S1 'AAb'
.SETS S2 'AA'
.SETS S3 'BBBB'
.IF S1 >= S2+S3[1:1] .INC N
```

Ker je nizovni simbol S1 večji od nizovnega simbola S2, ki smo mu dodali prvi znak od S3 (AAb >= AAB), se N poveča za ena.

6.3.5.2 Testiranje logičnih simbolov - direktivi .IFT in .IFF

.IFT/.IFF Direktivi testirata ali ima simbol vrednost TRUE (.IFT) ali FALSE (.IFF). Če je izid testa pozitiven, se izvede preostanek ukazne vrstice. Procesor javi usodno napako, če je bil simbol definiran kot numerični ali nizovni.

Oblika ukaza:

```
.IFT ssssss

.IFF ssssss
```

kjer ssssss pomeni ime simbola, ki ga testiramo. Pred in za imenom simbola mora biti vsaj en prazen znak.

Primera:

```
.IFT A .GOTO 100

.IFF A .GOTO 200
```

6.3.6 DELO Z PODATKOVNIMI DATOTEKAMI

Na voljo imamo tri direktive za delo z datotekami. Datoteke lahko samo kreiramo in pišemo v njih, najprej datoteko odpremo, pišemo vanjo in jo zapremo. Istočasno lahko imamo odprte 4 datotke.

6.3.6.1 Odpiranje datotek -

6.3.6.1.1 Direktiva .OPEN

Direktiva .OPEN odpre sekundarno datoteko s specificiranim imenom tipa .DAT kot izhodno datoteko. Vsakokrat se kreira nova datoteka.

Oblika ukaza:

.OPEN [#N] ime
(oklepaji niso del sintakse)

kjer je:

ime ime datoteke, ki bo odprta kot izhodna datoteka. Privzeti tip datoteke je .DAT.

n neobvezna številka datoteke; n je lahko od 0 do 3. V istem trenutku lahko imamo odprte štiri datoteke.

Primer:

.OPEN SEKUNDARNA

Direktiva odpre sekundarno datoteko SEKUNDARNA.DAT kot izhodno.

6.3.6.1.2 Direktiva .OPENA

Direktiva odpre obstoječo sekundarno datoteko s specificiranim imenom in doda vse podatke, ki sledijo.

Oblika ukaza:

.OPENA [#N] ime
(oklepaji niso del sintakse)

kjer je:

ime ime datoteke, ki bo odprta za dodajanje podatkov. Privzeti tip datoteke je .DAT.

n neobvezna številka datoteke; n je lahko od 0 do 3. V istem trenutku lahko imamo odprte štiri datoteke. število n lahko zamenjamo z numeričnim simbolom med narekovaji.

Primer:

.OPENA SEKUNDARNA

Direktiva odpre sekundarno datoteko SEKUNDARNA.DAT kot izhodno za dodajanje in doda vse podatke, ki sledijo.

6.3.6.2 Pisanje v datoteko -

V odprto datoteko lahko pišemo z direktivama .DATA in .ENABLE DATA. Direktiva .DATA zapiše eno vrstico v datoteko. Če želimo zapisati več vrstic, uporabimo direktivi .ENABLE DATA in .DISABLE DATA.

6.3.6.2.1 Direktiva .DATA -

Tekst za direktivo .DATA se prenese v podatkovno datoteko, ki smo jo odprli z .OPEN direktivo.

Oblika ukaza

```
.DATA [#n] tekst
(oklepaji niso del sintakse)
```

kjer je:

tekst tekst, ki se bo prenesel v podatkovno datoteko

n neobvezna številka datoteke od 0 do 3. Privzeta vrednost je 0. Številko lahko nadomestimo z simbolom med narekovaji.

Ukazna vrstica ne sme biti daljša od 80 znakov. Če podatkovna datoteka ni odprta, procesor obdela in javi napako

Primer:

```
.SETS A "TO SO PODATKI"
.OPEN TEMP
.DATA 'A'
.CLOSE
```

Te direktive prenesejo v datoteko TEMP.DAT tekst TO SO PODATKI.
A

6.3.6.2.2 Direktivi .ENABLE DATA in .DISABLE DATA -

Tekst med direktivama .ENABLE DATA in .DISABLE DATA se prenese v podatkovno datoteko. Direktiva .DISABLE DATA mora biti v prvi koloni.

Oblika direktive:

```
.ENABLE DATA [#n]

.DISABLE DATA [#n]
(oklepaji niso del sintakse)
```

n neobvezna številka datoteke od 0 do 3; privzeto 0. Zamenjamo jo lahko s simbolom med narekovaji.

Primer:

```
.OPEN PODATKI
.ENABLE DATA
      .
      .
      .
.DISABLE DATA
```

Vse, kar je med .ENABLE DATA in .DISABLE DATA, se prepiše v datoteko PODATKI.DAT.

6.3.6.3 Zapiranje datoteke - direktiva .CLOSE -

Na koncu pisanja moramo datoteko zapreti. Zapremo jo z direktivo .CLOSE.

Oblika ukaza:

```
.CLOSE [#n]
```

n neobvezna številka datoteke me 0 in 3. Privzeta vrednos je 0.

6.3.7 DIREKTIVE .ENABLE IN .DISABLE

Direktiva .ENABLE vspostavlja enega od šestih načinov delovanja: ločevanje malih in velikih črk, delo z podatki, globalne simbole, prepoznavanje <ESC> znaka, kontrolo odziva na ekranu. Vsak od načinov je neodvisen od ostalih; vsi so lahko hkrati aktivni. Ko startamo AT. procesor je aktiven le .ENABLE LOWERCASE. Če jih želimo vzpostaviti, moramo uporabiti direktivo .ENABLE način. Ogledali si bomo le tri načine.

Direktiva .ENABLE SUBSTITUTION nam omogoča zamenjavo imena simbola z njegovo vrednostjo. Simbol mora biti med enojnima narekovajema ('SIMBOL'). Če smo naprimer simbolu A priredili niz TO JE TEST, potem se vsak 'A' zamenja z TO JE TEST. Če je vspostaljen način substitucije, procesor izvrši zamenjavo, preden izvede direktivo ali MCR ukaz. (Ko izvede .GOTO direktivo, ne preverja ali so simboli med samo direktivo in ciljno vrstico, ki jo labela označuje, definirani.)

Direktivo .ENABLE DATA smo že spoznali.

Direktiva .ENABLE QUIET prepreči izpis MCR ukazov na ekran v času izvajanja posredne ukazovne datoteke.

Oblike direktiv:

```
.ENABLE SUBSTITUTION

.ENABLE QUIET
```

Primeri:

```
.ENABLE SUBSTITUTION
.ASKS PROG VNESI IME PROGRAMA
MAC 'PROG'='PROG'

Ko se procedura izvaja, se na terminal izpiše:

>*VNESI IME PROGRAMA [S]:TEST
>MAC TEST=TEST

.ENABLE QUIET
ACT
TIM

Ko izvajamo to proceduro, se izpiše

MCR...
...MCR
...AT.
15-NOV-83 14:15:55
```

6.3.7.1 Direktiva .DISABLE -

Direktiva prekliče način delovanja AT. procesorja, ki sa navedemo in ki smo jih prej vpostavili z direktivo .ENABLE.

Oblike ukaza:

```
.DISABLE SUBSTITUTION

.DISABLE QUIET
```

6.3.8 DIREKTIVE ZA KONTROLO IZVAJANJE

6.3.8.1 Direktiva .DELAY -

Direktiva .DELAY zadrži izvajanje ukazovne datoteke za navedeno število časovnih enot.

Oblika ukaza:

.DELAY nnu

kjer je

nn število časovnih enot zadrževanja

u enota, ki je lahko

T-ticksov
S-sekund
M-minut
H-ur

Parameter nn je osmiški po privzetku in desetiški, če je s piko.
Ko se direktiva izvaja, procesor sporoči:

AT. -- DELAYING

Ko čas poteče in procesor nadaljuje z delom, javi

AT. -- CONTINUING

Primer:direktiva

.DELAY 20M

zadrži izvajanje ukazovne datoteke za 20(8) oz. 16(10) minut.

6.3.8.2 Direktiva .PAUSE -

Direktiva .PAUSE prekine izvajanje posredne ukazovne datoteke in počaka intervencijo uporabnika. Procesor prekine sam sebe; uporabnik izvrši poseg in izda ukaz za nadaljevanje izvajanje AT..

Oblika ukaza:

.PAUSE

Ko se procesor sam prekine, izpiše sporočilo

AT. --PAUSING. TO CONTINUE TYPE "UNS tttttt"

kjer je tttttt ime taska za izvajanje posrednih ukazovnih datotek. Da bi se izvajanje nadaljevalo, moramo izdati ukaz

>UNS tttttt

AT. izpiše sporočilo

AT. --CONTINUING

in nadaljuje izvajanje posredne ukazovne datoteke.

6.3.8.3 Direktiva .WAIT -

Direktiva .WAIT prekine izvajanje posredne ukazovne datoteke dokler ne konča z delom specificirani task.

Oblika ukaza:

.WAIT [tttttt]

Ime taska mora ločevati od direktive vsaj en prazen znak. Če imena taska ne navedemo, AT. privzame ime taska, ki smo sa nazadnje poslali na izvajanje z MCR ukazom RUN. Task ima ime TTnn, kjer je TTnn ime in številka terminala. Če navedeni (ali privzeta vrednost) task ni instaliran, AT. direktive .WAIT ne upošteva. Če smo navedli stikalo /NOMCR, direktiva .WAIT nima učinka.

Primer:

.WAIT PIP

Izvajanje posredne ukazovne datoteke se prekine do trenutka, ko konča z delom PIP.

6.3.8.4 Direktiva .XQT -

AT. procesor navadno preda ukaze MCR-u in čaka, da se izvedejo. Vendar obstaja tudi možnost, da pošljemo task na izvajanje in ne čakamo, da se izvajanje konča, ampak izvajamo nadalnje direktive ukazovne datoteke. Z RUN ukazom lahko sprožimo izvajanje nekega taska in nato nadljujemo z izvajanjem posredne ukazovne datoteke, vendar ne moremo z RUN ukazom dodati tudi ukazovnega niza za navedeni task. Direktiva .XQT nam omogoča inicializacijo taska tudi z ukazovno vrstico. Ko procesor izvede direktivo, preide takoj na sledečo vrstico ukazovne datoteke. AT. dovoljuje do 10 (10) zaporednih .XQT direktiv.

Oblika ukaza:

.XQT ime-taska ukazovni-niz

kjer je ime-taska ime taska, ki sa inicializiramo in ukazovni-niz ukazovni niz k temu tasku.

Direktiva .XQT nam omogoča vporedno izvajanje taskov. Z direktivo .WAIT lahko nato usklajujemo izvajanje vporednih taskov; izvajanje ukazovne datoteke se prekine, dokler navedeni task ne konča z delom.

Primer:

```
.XQT MAC TEST,TEST=TEST
.XQT TKB BLD,BLD=BLD
.WAIT MAC
.WAIT TKB
```

Ta primer ukazovne datoteke sproži asembliranje in TKB vzporedno in potem čaka, da oba taska končata z delom.

V A J E

1. Napiši posredno ukazovno datoteko za razvoj fortranskega programa.

2. Napiši posredno ukazovno datoteko, ki v sistemskem datumu zamenja ime meseca z številko meseca in ta datum izpiše na ekran.

3. Napiši posredno ukazovno datoteko, ki te pozdravi 'DOBRO JUTRO', če jo startaš pred deveto uro, 'DOBER DAN' če jo stratáš med deveto in osemnajsto in 'DOBER VECER' če jo startaš po osemnajsti uri.

(1)
```
.XQT  FOR  PR, PR=PR
.XQT  TKB  PR=PR
.WAIT MAC
.WAIT TKB
.STOP
```

POGLAVJE 7

IZPISOVANJE DATOTEK NA PRINTERJU

Izpis na tiskalniku dosežemo na več načinov. Dva načina smo že spoznali, ko smo obravnavali PIP pomožni program. Tukaj bomo spoznali še en način izpisa.

7.1 UKAZ "PRINT"

Ukaz PRINT uvrsti navedeno datoteko v čakalno vrsto Queue Managerja, ki poskrbi za izpis na tiskalnik. Queue Menager ali urejevalnik čakalne vrste za izpis lahko nadzoruje več čakalnih vrst na večih tiskalnikih. Vrstni red izpisa datotek, ki čakajo na izpis je odvisen od večih kriterijev.

Da računalnik lahko sprejme ukaz PRINT, mora biti v sistemu prisoten sistemski program Queue Manager. Ker vse računalniške instalacije tega programa ne vključujejo, naj uporabnik sam preveri listo instaliranih taskov v njegovem sistemu (TAL ukaz). Če sta instalirana taska QMG...in PRT...pomeni, da sistem vsebuje program Queue Manager, če se pojavi v listi le PRT...vsebuje sistem le program za navadni serijski izpis. V tem primeru lahko uporabimo za izpis datotek na tiskalniku le PIP s kretnico /SP.

Če nobeden od omenjenih taskov ni instaliran, se lahko datoteke izpisujejo le direktno z ukazom PIP LP0:=specif. datot. Uporabnik v tem primeru nima prisvojenega tiskalnika. Če med njegovim izpisovanjem izda enak ukaz drug uporabnik, bosta njuna izpisa na papirju pomešana med seboj.

Če ste slede statusa vašega sistema v kakršnihkoli dvomih, se posvetujte z vašim sistemskim vodstvom. Ukaz PRINT v najenostavnejši obliki:

```
>PRI spec.dat.[/CO:n],spec.dat.[/CO:n],..
```

izvrši vpis navedenih datotek v datoteko SY:[1,7] QUEUE.SYS. Ta vpis ima v vrsti svojo številko in ime, ki sa tvori prvih šest znakov imena prve navedene datoteke. I Vrsta ima privzeto ime "Print". Izpis se izvrši takoj, ko je vrstični tiskalnik prost. Med pisanjem si tiskalnik prisvoji izpisni program, da ne more priti do mešanja izpisov. Ko je izpis končan, se vpis v datoteki vrste na [1,7] izbriše. Vsebina čakalne vrste se lahko izpie na

terminalu z ukazi QUE. Če navedemo še kretnico /CO:n povemo Queue Menaserju, da želimo n kopij navedene datoteke.

Ukaz PRINT sprejme večkratne specifikacije datotek, pa tudi zvezdice. Izpis se izvrši v zaporedju, danem v ukazu. Če so datoteke podane z zvezdicami, se izpisejo v takem zaporedju, kot se pojavljajo v UFD-ju (izpis z ukazom PIP/LI).

Z ukazom PRINT določimo nadzorniku vrste 'izpisni posel'. Ta posel lahko vsebuje eno ali več datotek, ki jih je treba izpisati. Z raznimi določili lahko določimo različne posebnosti izpisnega posla:

- čas, ko naj se spool izvrši
- enoto, ki naj izpise specificirano datoteko
- prioriteto (specificno za vrsto) izpisnega posla
- ponovljivost posla
- formular na katerega naj se piše
- število vrstic na strani
- število kopij posamezne datoteke
- brisanje datoteke po izpisu

Teh določil tukaj ne bomo obravnavali. Uporabnik lahko z neposrednimi ukazi nadzorniku vrste dobi na zaslonu prikaz čakalnih vrst ali posameznih izpisnih poslov. Posel je možno zadržati v vrsti, sa sprostiti, ali umakniti iz vrste. Da bi imeli pristop do nadzornika vrste, le-tega ni treba eksplicitno pozivati. To stori avtomatično ukaz PRINT. Seveda lahko damo informacije nadzorniku vrste tudi neposredno z ukazom QUE. Nadzornik vrste se ne odziva eksplicitno kot drugi pomožni sistemski programi in sprejema le enovrstične ukaze.

Nekateri sistemski programi, ki proizvajajo izpisljive informacije (prevajalniki, zbirnik MACRO-11, povezovalnik binarnih modulov - Task Builder), ki se jih poziva neposredno iz MCR-a, imajo svoja pravila za izpis proizvedenih datotek (spool). Če eksplicitno zahtevamo izpis (spool) kake take datoteke, to izvrši nadzornik vrste.

Če je v sistemu instaliran namesto nadzornika vrste program za serijski despool, vsi programi, ki proizvajajo izpisljive datoteke, delujejo kot običajno. Če niti ta ni instaliran, potem se te datoteke, ki nastanejo na disku, ne izpišejo na pisalniku. Izpisni posli se lahko uvrstijo v čakalno vrsto tudi s programom PIP z naslednjim ukazom:

```
>PIP DATOTEKA.TIP/SP.
```

Izpisni posel dobi ime, ki sa tvori prvih šest črk imena prve datoteke v poslu.

7.2 UKAZNI NIZ ZA NADZORNIKA VRSTE (QUEUE MANAGER)

Ukazi nadzorniku vrste imajo naslednjo obliko:

>QUE identifikacija posla /določilo [:opcija]

Identifikacija posla je lahko ime, ki ga ima posel, ali pa vhodno število, ki ga nadzornik vrste podeli poslu ob vključitvi v vrsto. To število se prikaže z ukazom QUE/LI. Njegova oblika je (n1.n2), v ukazih se navaja v obliki JOB:n1: n2.

Ukazi QUE omogočajo neprivilegiranemu uporabniku naslednje:

- Izpis vhodnih točk vrste (poslov v vrsti) v različnih oblikah.
- Spreminjanje atributov poslov, ki čakajo v vrsti.
- Zadrževanje posla v vrsti.
- Sprostitev zadržanih poslov.
- Brisanje poslov iz vrste.

Nadzornik vrste dopušča le enovrstične ukaze QUE.

7.2.1 UKAZ QUE/LIST

Ukaz QUE/LIST da pregled poslov v vrsti. Če ne specificiramo opcij, ali ne imenujemo izhodne enote, se izpišejo na ekranu le posli, vključeni v osnovni vrsti PRINT. V izpisu je navedeno tudi kateri izhodni enoti pripada posamezna vrsta.

Oblika ukaza:

>QUE specifikacija posla /LI

Namesto /LI lahko uporabimo tudi /BR ali pa /FU. Prvi nam daje manj informacij kot /LI kretnica, drugi pa več. Če imena posla ne navedemo, nam izpiše informacije o vseh poslih v PRINT čakalni vrsti.

Identifikacija posla: Posel identificiramo po njegovem imenu, ki mu spredaj dodamo še UIC, če se ne nahajamo na istem UIC-ju, s katerega je bil posel poslan v izpis. Lahko pa posel identificiramo tudi po dveh številkah, ki jih Queue Menager dodeljuje sam. Identifikacijo navedeno kot kretnica /JOB:n1:n2 namesto specifikacije v obliki imena. Če želimo izpisati vse posle z istim UIC-jem navedemo samo [UIC]/kretnica za izpis (/LI,/BR,/FU).

/BRief Skrajšani izpis čakalne vrste vsebuje samo imena poslov, UIC, s katerega je bil posamezen posel izdan, vhodno številko posla in status posla. Posel je lahko aktiven (ACTIVE), zadrzan (HELD), časovno blokiran (PRINT AFTER) ali čaka, da pride na vrsto (brez označbe).

/LIst Standardni izpis čakalne vrste vsebuje imena poslov, UIC, s katerega je bil posel izdan, vhodno stevilko posla, status posla in vse datoteke, ki v poslu čakajo na izpis.

/Ful Popoln izpis čakalne vrste vsebuje vse tisto, kar je standardni izpis, poleg tega pa še vse eksplicitno določene in avtomatično privzete atribute posla.

Primeri:

```
>QUE /LIST
**PRINT QUEUES**
PRINT=>LP0:
[100,57] DUMP    (2200,144)                    ACTIVE ON LP0:
         DE0:[100,57]DUMP.MAC;1                COP:5
[300,300]X       (2300,146)
         DE1:[300,300]X.;26
[100,62] AAA     (2400,150)
         DE0:[100,62]AAA.LST;1
[100,62] AAA     (2500,152)
         DE0:[100,62] AAA.LST;2
[100,57] FLOPPY  (2000,140)                    HELD
         DE0:[100,57]FPY.MAC;1
         DE0:[100,57]CLR.MAC;1
         DE0:[100,57]FZ.MAC;1                  COP:2
```

Izpis obsega vse posle v vrsti PRINT v standardnem formatu. Izpisane so tudi vse datoteke, ki sestavljajo posamezen posel. Posel DUMP je trenutno aktiven, FLOPPY je blokiran. Preostala posla čakata, da prideta na vrsto.

```
>QUE /FU
**PRINT QUEUES**
PRINT=>LP0:
[7,107] FLOPPY  (2500,56)        FORM:0          ACTIVE ON LP0:
        PRI:50  LEN:65  PAGE:0   NORESTART FLAG
        DE0: [100,57] FPY.MAC;5  COP:1   NODELETE
        DE0: [100,57] F.MAC;10   COP:1   NODELETE
        DE0: [100,57] FZ.MAC;1   COP:3   NODELETE
[7,107] DUMP    (2200,62)        FORM:0  PRINT AFTER 31-JUL-80 13:00
        PRI:50  LEN:65  PAGE:0   NORESTART FLAG
        DE0: [100,57] DUMP.MAC;1         COP:12  NODELETE
```

Izpiše se vrsta PRINT. Razširjeni format obsega še vse atribute poslov. Posel DUMP se bo aktiviral šele po 13. uri.

```
>QUE /BR
**PRINT QUEUES**
PRINT=>LP0:
[100,57] DUMP    (2200,144)      ACTIVE ON LP0:
[300,300] X      (2300,150)      PRINT AFTER 31-JUL-80 16:30
[100,62] AAA     (2400,150)
[100,62] AAA     (2500,152)
[100,57] FLOPPY  (2000, 140)     HELD
```

V skrajšanem izpisu vidimo le posle in njihov status.

```
>QUE DUMP /FU
```

```
**PRINT QUEUES**
PRINT=>LP0:
[7,107] DUMP     (2200,62)        FORM:0  PRINT AFTER 31-JUL-80
        PRI:50  LEN:65  PAGE:0  NORESTART FLEAG 13:00
                DE0: [100,57]DUMP.MAC;1 COP:12  NODELETE
```

Dobili smo popolno informacijo o poslu DUMP iz vrste PRINT.

```
>QUE[100,57]/BR
**PRINT QUEUES**
PRINT=>
[00,57] DUMP     2200,144
[00,57] FLOPPY   2000,140        HELD
```

Dobimo skrajšan izpis poslov v vrsti PRINT, ki so bili kreirani iz navedenega UIC-ja.

```
>QUE /JOB:2000:140/LI/FU
**PRINT QUEUES**
PRINT=>LP0:
[100,57]  FLOPPY         (2000,140)      FORM:0  HELD
        PRI:50  LEN:65  PAGE:0  NORESTART NOFLAG
        DE0:[100,57] FPY.MAC;5  COP:1   NODELETE
        DE0:[100,57] CLR.MAC;1  COP:1   NODELETE
        DE0:[100,57] FZ.MAC;1   COP:2   NODELETE
```

Namesto z imenom, lahko posel identificiramo tudi z njegovim enoličnо prirejenim vhodnim številom.

7.2.2 UKAZ QUE/DELETE

Z ukazom QUE/DELETE izbrišemo posel iz čakalne vrste. S tem ukazom ne moremo brisati datotek iz direktorijev. Ukaz namreč nima nikakršne zveze z ukazi PIP/DE ali PRINT/DE.

Oblika ukaza:

```
>QUE identifikacija posla /DELETE
```

Identifikacija posla: Posel lahko identificiramo na dva načina:

[UIC]ime posla, če je posel iz UIC-ja, na katerem smo prijavljeni, zadostuje za identifikacijo ime posla. UIC moramo obvezno napisati v oglatem oklepaju.

/JOB:n1n2

Vhodno številko priredi poslu nadzornik vrste, ko ga uvrsti v vrsto. Dobimo jo v izpisu vrste v obliki (n1,n2). Določilo /DELETE specificira, da se imenovani posel izbriše iz vrste. Če se v trenutku, ko izdamo ukaz, specificirani posel izpisuje, se izpis prekine in se posel izbriše iz vrste.

Primer:

```
>QUE PETER/DELETE
```

Posel PETER iz UIC-ja, na katerem smo prijavljeni, se izbriše iz vrste PRINT.

POGLAVJE 8

PROGRAM ZA PREPISOVANJE DATOTEK (FLX)

Program za prepisovanje datotek (FLX) uporabljamo za prepisovanje datotek z ene enote na drugo. Pri prepisovanju datoteke z enote, ki ima različen format zapisa od enote, na katero se datoteka prepisuje, FLX pretvarja format datoteke v tako obliko, ki odgovarja enoti, na katero se datoteka prepisuje. S programom za prenos datotek je možno:

- izpisovanje direktorijev na kasetah, enotah RT-11 ali DOS-11
- brisanje datotek z organiziranih enot RT-11 ali DOS-11
- inicializiranje kaset, enot RT-11 ali DOS-11.

S FLX-om lahko prepisujemo datoteke (in spreminjamo format) z:

- DOS-11 na enote FILES-11
- FILES-11 na enote DOS-11
- DOS-11 na enote DOS-11
- FILES-11 na enote FILES-11
- FILES-11 na enote RT-11
- RT-11 na enote FILES-11
- RT-11 na enote RT-11

Veljavne enote DOS-11 so:

Okrajšava enote:	Enota:
CT	kasete TU60
DK	disk RK05
DT	trdna enota TU56 DEC
MM	magnetni trak TE16, TU16, TU45, ali TU77
MS	magnetni trak TS04
MT	magnetni trak TU10 ali TS03
PP	luknjalec traku PC11
PR	čitalec papirnega traku PC11 ali PR11

Veljavne enote RT-11 so:

Okrajšava enote:	Enota:
DD	trak TU58 DEC
DK	disk RK05
DL	disk RL01

DM	disk RK06 ali RK07
DT	tračna enota TU56 DEC
DX	disketa RX01
DY	disketa RX02

FLX podpira vse enote FILES-11, vključno s kasetami formata RSX. Enote FILES-11 so privzete enote, katere smo inicializirali z MCR ukazom INITVOL. Enote DOS-11 in RT-11 inicializiramo s FLX. Kretnice za formate so naslednje: /RS za format FILES-11, /DO za format DOS-11 in /RT za format RT-11. Te kretnice so opisane v naslednjem podposlavju.

FLX lahko kličemo na dva načina; tako da specificiramo FLX in ukaz ali da samo specificiramo FLX. V tem primeru vam program odgovori s promptom:

```
>FLX
```

8.1 FLX UKAZNI NIZ

Oblika ukaza za kopiranje z FLX je:

```
enota/kretnica(e)=vh.dat1/kretnica(e),...,vh.datN/kretnica(e)
```

kjer so:

enota

je specifikacija izhodne enote FLX v obliki: enota:ufd. Polje za UFD ni obvezno; če UFD ni specificiran, uporabi FLX trenutni UIC. Kadar specificirate izhodno enoto, morate navesti tudi enačaj. FLX ne dovoljuje specifikacije izhodnih datotek. Izhodne datoteke privzamejo imena vhodnih datotek.

vh.datN

je specifikacija vhodne datoteke v obliki enota:ufd ime.tip;verzija

/kretnica

je en od treh tipov kretnic FLX, ki so opisani v naslednjem podposlavju.

FLX dovoljuje 9 znakov za imena datotek pri magnetnih trakovih DOS-11. Zvezdice so dovoljene za specifikacijo vhodne datoteke. Številke verzije so dovoljene le za datoteke tipa FILES-11 in ne smejo biti specificirane z zvezdicami.

8.2 KRETNICE FLX

FLX ima tri tipe kretnic za prepisovanje datotek:

- kretnice za formatiranje
- kretnice za prepisovanje
- kontrolne kretnice

S kretnicami za formatiranje specificiramo format enote na kateri so datoteke; to pomeni enote tipa FILES-11, DOS-11 ali RT-11. S kretnicami za prepisovanje specificiramo način prenosa, ko datoteka nima standardnega tipa. Datoteke je možno formirati v ASCII, binarni ali v formatu slike datoteke.

Kontrolne kretnice omogočajo kontrolne funkcije v času prenosa datoteke. Lahko specificirate npr. število blokov, ki naj se rezervirajo za izhodno datoteko, ali specifirate UFD za izhodno datoteko.

8.2.1 KRETNICE ZA FORMATIRANJE

FLX ima tri kretnice za definicijo formata specificirane enote.

Tabela kretnic
FLX-sovih formatov

Kretnica	Opis
/DO	identifikacija formatirane enote DOS-11
/RS	identifikacija formatirane enote FILES-11
/RT	identifikacija formatirane enote RT-11

Na začetku je privzeta vrednost za vhodne enote DOS-11 format in FILES-11 format za izhodne enote. FLX uporablja te privzete vrednosti, če ne navedete kretnic v ukazu, vendar je priporočljivo uporabljati format kjuče zaradi večje jasnosti, kako stukturo ima določena enota.

8.2.2 KRETNICE ZA PREPISOVANJE

FLX ima tri kretnice za prepisovanje datotek, po enega za vsak tip formata datoteke. Datoteke so lahko formatirane ASCII, formatirane binarno ali v formatu slike datoteke. Pretvorbe formatov so možne v vseh smereh med datotekami DOS-11 in FILES-11. S specifikacijo kretnice za prepisovanje se določi, kakšen bo format izhodne datoteke po njeni konverziji.

Tabela kretnic
za prepisovanje FLX

Kretnica	Opis
/FA:n	formatirano ASCII Izhodne datoteke DOS-11 in RT-11 morajo biti formatirane ASCII. To pomeni, da so zapisi ASCII podatki zaključeni s carriage return/nova vrstica (CR-LF), form feed (FF), ali vertical tab (VT). V prepisovanju datotek iz DOS-11 ali RT-11 v datoteke FILES-11 so pari CR-LF izločeni iz zapisov. V prepisovanju iz FILES-11 v DOS-11 ali datoteke RT-11 so pari CR-LF dodani na konec vsakega zapisa, ki ni zaključen z LF ali s FF. V obeh smereh so iz vhodnih zapisov izločene vse ničle, rubouts in vertical tabs. Če specificirate /FA:n na izhod FILES-11 se generirajo fiksni zapisi dolžine n. Kadar je potrebno, so izhodni zapisi dopolnjeni z ničlami. Če ne specificirate /FA:n na izhodni strani FILES-11 se generirajo zapisi variabilne dolžine tam, kjer je izhodni zapis enak vhodnemu zapisu. Podatki ASCII se prenašajo kot 7 bitne vrednosti. Bit 8 se maskira pred prenosom. CTRL/Z (ASCII 032 oktalno) pomeni logični konec vhodne datoteke za formatirano ASCII prepisovanje iz kaset ali trakov DOS-11 v FILES-11.
/FB:n	Formatirano binarni Izhodne datoteke DOS-11 ali RT-11 morajo biti formatirane binarno. To pomeni, da se izhodnim zapisom DOS-11 ali RT-11 dodajajo formatirane binarne slave in paritete, pri prenosih v datoteke FILES-11 pa se izlocijo. Če specificirate /FB:n se tvorijo pri FILES-11 izhodu zapisi fiksne dolžine n (512 bytov decimalno je največja dolžina). FLX doda ustrezno število ničel pri tvorbi fiksnih zapisov. Če /FB:n ne specificirate pri izhodu FILES-11, se tvorijo zapisi variabilne dolžine tam, kjer je dolžina izhodnega zapisa enaka dolžini vhodnega zapisa.
/IM:n	Format slike datoteke (Image Mode) Prenos mora biti v formatu slike datoteke, kjer dobimo zapise fiksne dolžine. Dolžina zapisa je določena z vrednostjo n (v znakih decimalno) in je za izhod FILES-11 največ 512 znakov. Če ne navedete, privzame FLX vrednost 512 desetiško.

FLX uporablja naslednje privzete vrednosti za prepisovanje datotek različnih tipov

NAČIN	KRETNICA	TIP DATOTEKE
format slike datoteke	/IM:n	.TSK,.OLB,.MLB,.SYS,.SML,.ULB
formatirano binarno	/FB:n	.OBJ, .STB, .BIN, .LDA
formatirano ASCII	/FA:n	vsi ostali

FLX ignorira vrednost n, če jo specificirate pri /IM, /FB, /FA, kadar izhodna datoteka ni tipa FILES-11.

8.2.3 KONTROLNE KRETNICE

FLX ima več kontrolnih kretnic za kontrolo procesiranja datotek.

Tabela
kontrolnih kretnic FLX

Kretnica	Opis
/BL:n	Pomeni, da naj se n sklenjenih blokov rezervira za izhodno datoteko. Ta kretnica običajno uporabljamo s kretnico /CO (opisan spodaj). Če kretnice /BL ne navedete, bosta dolžini vhodne in izhodne datoteke enaki. Pri enotah RT-11 se za novo datoteko rezervira največji razpoložljivi prostor na enoti. Pri uporabi krtnice /BL:n s kretnico RT za izhodno datoteko se za izhodno datoteko rezervira prvi prosti prostor dolžine n, vendar ko se datoteka RT-11 zapre, sta dolžini vhodne in izhodne datoteke enaki. Če vhodna datoteka ni dolžine n, se pojavi napaka. Ker so vse datoteke RT-11 zapisane zaporedno, ni potrebno uporabiti kretnice /CO s kretnico BL:n, če gre za izhodno datoteko RT-11.
/BS:n	S tem določimo dolžimo bloka (n) v znakih za izhod pri kasetnih trakovih. Kadar kretnice /BS ne navedete, mislimo s tem blok dolžine 128(10) znakov. /BS se sme uporabljati le pri specifikaciji izhodne datoteke na kaseti (CT) skupaj s kretnico /RS.
/CO	Pomeni, da naj bo izhodna datoteka zapisana v zaporednih blokih. Kretnico /CO uporabljamo samo pri diskih in trakovih DEC. Kadar je vhodna datoteka papirni trak, kaseta ali magnetni trak

DOS-11, je potrebno specificirati tudi /BS. FLX prenaša datoteke tipov .TSK, .SYS in .OLB na enote FILES-11 s /CO implicitno, kadar je vhod enota FILES-11 ali pa trak ali disk DOS-11 ali RT-11 DEC.

/DE — Brišejo se datoteke z traku ali diska DOS-11 DEC. Uporablja se tudi z /RT za brisanje datotek s traku RT-11 DEC ali diska. Ko specificirate /DE ukaz, FLX nima izhodne specifikacije.

/DI — Izpise se direktorij kasete ali enot DOS na specificirano izhodno datoteko. Uporablja se tudi z /RT za generiranja izpisa direktorija enote RT-11 na specificirano izhodno datoteko. Enot FILES-11 ni možno izpisati s FLX. Če ne specificirate izhodne enote, se direktorij izpiše na TI:.

Če ne specificirate imena in tipa datoteke na vhodni strani, se privzame vrednost *.*.

/DNS:n — S tem specificiramo gostoto magnetnega traku; n je 800 ali 1600. Če n ni specificiran, ali pa ima kakšno drugo vrednost, javi FLX napako. Če kretnice /DNS:n ne navedete, se privzame vrednost 800. Če specificirate /DNS za enoto, ki ni magnetni trak, FLX ignorira to kretnico.

/FC — Pri uporabi datotek FORTRAN pomeni ta kretnica da se uporabljajo simboli FORTRAN carriage control; to pomeni, da v datotekah FORTRAN interpretira FORTRAN določene simbole, kot carriage kontrolne simbole. Kretnica FC se uporablja le za izhodne datoteke FILES-11 .

/ID — Izpiše se številka verzije FLX-a. To kretnico lahko specificirate kot del vhodne ali izhodne specifikacije, ali kot odziv na prompt FLX.

/LI — enako kot DI.

/NU:n — Uporablja se s kretnicami /ZE in /RT za specifikacijo števila blokov direktorija (n), ki naj se rezervirajo pri inicializaciji diska RT-11 ali traku DEC. Če kretnice ne navedemo, se rezervirajo štirje bloki. Največje število blokov, ki jih lahko rezerviramo je 31 desetiško, oziroma 37 osmiško.

/RW — Pred začetkom prepisovanja se previje magnetni trak. Če navedemo /-RW začne FLX prepisovanje prejšnjega previjanja traku. Če kretnice ne

navedete, se privzame vrednost /RW. Če sa navedete za enoto, ki ni trak, ali s kretnico /LI, /DE ali /ZE sa FLX ignorira.

/SP Pomeni, da naj se konvertirana datoteka tiska na vrsticnem tiskalniku s pomočjo print spoolerja ali queue manaserja. To kretnico lahko uporabljamo le za izhodne datoteke FILES-11.

/UI Pomeni, da naj ima izhodna datoteka enak UFD kot vhodna datoteka. FLX ignorira to kretnico če je UFD eksplicitno naveden. Kretnico VE je možno uporabiti le s kretnico CT pri specifikaciji izhodne datoteke.

/ZE Uporablja se za inicializacijo kaset ali enot DOS-11. Uporablja se tudi z /RT (in /NU) kretnico za inicilizacijo enot RT-11. Kretnica ZE ne zahteva specifikacije datoteke.

8.3 DELO Z DIREKTORIJI DOS-11

V tem poglavju so primeri, kako prikažemo direktorij DOS-11, kako brišemo datoteke DOS-11 in inicializiramo enote DOS-11 z uporabo kretnic FLX.

8.3.1 PRIKAZ DIREKTORIJEV DOS-11

S kretnico LI ali DI izpišemo direktorij kasete ali enote DOS-11 na datoteko FILES-11, katero specificiramo na izhodni strani. V kolikor ne specificirate izhodne datoteke, poslje direktorij FLX na TI:. Na primer:

```
FLX>DT0:[100,100]*.MAC/LI
```

S tem ukazom izpišemo na našem terminalu direktorij vseh datotek tipa .MAC na UIC 100,100 na traku DOS-11 DEC na DT0:.

Izpis direktorija traku DEC

```
DIRECTORY       DT: [200,200]
23-JUL-80

FLX.TSK          106.          16-JUL-80<233>
UFD.TSK           10.          16-JUL-80<233>
TKN.TSK            8.          16-JUL-80<233>
MOU.TSK           17.          16-JUL-80<233>

      TOTAL OF 141. BLOCKS IN 4.FILES
```

Izpis direktorija DOS-11

8.3.2 BRISANJE DATOTEK DOS-11

Datoteke z diskov DOS-11 ali s trakov DEC lahko brišemo s kretnico /DE. Pri uporabi te kretnice smemo specificirati le eno datoteko. Na primer:

```
FLX>DK1:[100,100]SYS1.MAC/DE
```

S tem ukazom brišemo datoteko SYS1.MAC z UFD[100,100] z diska DOS-11 na DK1:.

8.3.3 INICIALIZACIJA ENOT DOS-11

Z uporabo kretnice ZE lahko inicializirate kasete in enote DOS-11, potrebno je le specificirati enoto, katero želite inicializirati. Na primer:

```
FLX>DT:/ZE
```

S tem ukazom inicializiramo trak DEC na DT1: v formatu DOS-11.

8.4 DELO Z DIREKTORIJI RT-11

V tem podpoglavju so opisani postopki za prikaz izpisov direktorijev RT-11, brisanje datotek RT-11 in inicializacijo enot RT-11 z uporabo kretnicFLX.

8.4.1 PRIKAZ DIREKTORIJEV RT-11

S kombinacijo kretnic LI ali DI in kretnice RT je možno izpisati direktorij enote RT-11 na datoteko FILES-11 na izhodni strani. Če ne navedete izhodne datoteke, se direktorij izpiše na TI:. Na primer:

```
FLX>DT0:*.MAC/LI/RT
```

S tem ukazom izpišete terminalu vse datoteke .MAC enote RT-11 na DT0:. Primer izpisa:

```
DIRECTORY       DK:
6-JUL-80

KRIP.MAC         49.      6-JUN-80
<UNUSED>          7.
KIK.MAC          12.      12-MAJ-80
KIKO.MAC          7.      12-MAJ-80
<UNUSED>         16.
KRIPO.MAC         6.       6-JUN-80
<UNUSED>       2376.

2399. FREE BLOCKS
TOTAL OF 74.BLOCKS IN 4. FILES
```

Izpis direktorija RT-11 RK05 disk

8.4.2 BRISANJE DATOTEK RT-11

Datoteke z diskov RT-11 ali s trakov DEC lahko brišemo z uporabo kretnice /DE skupaj s kretnico /RT. V ukazu je potrebno specificirati le datoteke, katere želimo brisati in kretnici /DE/RT. Na primer:

```
FLX>DK1:SAM.MAC/DE/RT
```

S tem ukazom smo zbrisali datoteko SAM.MAC z enote RT-11 na DK1:.

8.4.3 INICIALIZIRANJE ENOT RT-11

Enote RT-11 lahko inicializiramo s kretnico ZE skupaj s kretnico RT. Potrebno je specificirati le obe kretnice za inicializacijo, na primer:

```
FLX>DT1:/ZE/RT
```

S tem ukazom inicializiramo trak DEC na DT1: v formatu RT-11. Pri inicializaciji enot RT-11 lahko s kretnico /ZE uporabimo še argument v obliki

```
/ZE:n
```

Vrednost za n označuje število dodatnih besed za dostop do direktorija. Normalno je dostop do direktorija dolg sedem besed.

Z uporabo kretnice /NU skupno z /ZE in /RT se določi število sesmentov direktorija na enoti RT-11. Oblika kretnice /NU je sledeča:

```
/NU:n
```

Vrednost n pomeni koliko sesmentov direktorija naj se rezervira (alocira). Privzeta vrednost za n je štiri. Največje število sesmentov je 37(8) oziroma 31(10). Na primer:

```
FLX>DT0:/ZE:2/NU:6/RT
```

S tem ukazom inicializiramo trak DEC na DT0: v formatu RT-11, rezervirajo se po dve dodatni besedi za vsak dostop do direktorija in se alocira šest segmentov direktorija. Rezultat je, da dobimo 54(10) dostopov do direktorija, od katerih je vsak dolg 8 besed.

DODATEK A

LITERATURA IN VIRI

RSX-11M/M-PLUS MCR Operations Manual,
Order No.AA-H263A-TC, DEC, Maynard 1979

RSX-11 Utilities Manual
Order No.AA-H268A-TC, DEC, Maynard 1979

PDP11 Software Handbook, DEC, 1980

PDP11 Processor Handbook, DEC, 1980

računalniški sistemi delta

Izdajatelj:

ISKRA - DELTA, IZOBRAŽEVALNI CENTER DELTA, Ljubljana, Titova 51

Avtor:
Slavko Lenarčič

Ljubljana, september 1984