Unix’i loeng 17.04.97 1.

Programmid ja protsessid

argument - a
kui C P-i täidetakse kui käsku, siis käsureal olevad a-d on kättesaadavad funktsioonile ***main*** a-de loendi ***argc*** ja viida ***argv*** abil (viitab massiivile, mille elementideks on sümbolite massiivid, mis sisaldavad a-e, iga massiiv lõpeb \0-ga, s.t. neid võib vaadelda stringidena)
argv[0] - käsu enda nimi, seega alati ***argc*** > 0
järgmine P kuvab oma argumendid:

```
main (argc, argv)       /* argumentide kuva  eraldi ridadel   */
int argc;
char *argv[];
{
   int i;
   for (i=0; i<argc; i++)
      printf("%s\n", argv[i]);
}
```

argc - **arg**ument **c**ount argv - **arg**ument **v**ector
a-d lõppevad ‘\0‘ - s.t nad on stringid
S-ile antakse edasi P’i nimi ja viitade massiivi algusaadress
U’i T vastutab kõikide andmevahetuse operatsioonide eest RAM’i, P-de, F-de ja I/O-seadmete vahel
rakendusP-d saavad pöörduda süsteemsete vahendite poole nn. süsteemsete väljakutsete (pöördumiste) abil
vaatleme seda C baasil
paljud vahendid, mis on olemas teiste keeltes, on C-s realiseeritud standardfunksioonide abil; näiteks 2 stringi võrdlus
huvitav on ***printf,*** mis viib andmed väljundvoogu ja teeb vormindatud (formaaditud) väljastust

17.04.97 2.

standardsisend ja standardväljund

andmed suunduvad voogudena, s.t. baitide jadadena RAM-st F-desse (I/O seadmetele) ja vastupidi

P'i käivitamisel avatakse automaatselt 3 F'i (voogu):

stdin - standartne sisendF

stdout - standartne väljundF

stderror - standartne veaF

need 3 F'i (voogu) on vaikimisi seotud terminaliga, kuid neid võib ümber suunata

nende i-node'id on vastavalt 0, 1, 2

lihtsaim I - sisestus **stdin**-st, selleks funkts. ***getchar***, mis tagastab järgmise sümboli; võib sisestada ka F-st või teisest P-st:

$ prog < failinimi $ teinepr | prog

(prog pöördub ***getchar***-i poole)

getchar tagastab EOF, kui jõuab F'i lõppu (viga)

EOF on defineeritud kui -1

putchar(c) paneb sümboli 'c' **stdout**-i, võib ka F-i või teise P-i:

$ prog > failinimi $ prog | teinepr

printf viib andmed **stdout**-i, vormindab O

fprintf - identne ***printf***-ga, v.a 1. argument on F-viit

scanf loeb voost **stdin,** võimaldab I vormindada (jagada stringideks, numbriteks,…)

#include <stdio.h> - peab olema iga lähteF-i algul, mis teeb I/O ja kasutab standartseid I/O-funktsioone

tegelikult toimub U-s kogu I ja O läbi alumise taseme P-de ***read*** ja ***write;***

pöördumised:

n_read = read (fd, buf, n) ;

n_written = write (fd, buf, n) ;

fd - F-i i-node
buf - P-i puhver, kust andmed tulevad või kuhu lähevad
n - ülekantavate baitide arv
tagastatakse tegelikult ülekantud baitide arv
n väärtus harilikult 1(sümbolhaaval) või 1024(plokikaupa)
näide funktsioonist ***getchar***

```
# include <stdio.h>
# define CMASK  0377  /* teeb char'id > 0  */
getchar()           /*      puhverdamata 1 char-i   I */
{
      char c;
      return((read(0, &c, 1) > 0) ? c & CMASK : EOF);
}
```

&c - c aadress
& - bitthaaval loogiline korrutamine

F-d

siiani kõik funkts.-d opereerisid standardvoogudega
aga kui vaja I/O-operatsioone teha muude F-dega ?
F tuleb avada standardfunktsiooniga ***fopen***, mis pöördub OS-i poole ja tagastab nn. sisemise F-nime, mis tegelikult on viit (F-viit) struktuurile, kus on info F kohta
vajalik deklaratsioon:

```
FILE    *fp,  *fopen() ;
```

mis ütleb, et fp on viit 'FILE '-le ja et fopen tagastab viida 'FILE '-le
FILE - tüübinimi (nagu int)
tegelik pöördumine:

```
fp = fopen (nimi, olek)
```

mõlemad par.-d on stringid, nimi - F-nimi, olekud: "r", "w", "a", vastavalt read, write, append; olekule võib järgneda "+", mis tähistab F-i avamist lugemiseks **ja** kirjutamiseks

“r+“ - voog F-i algusest
“w+“ - tekitab või puhastab F-i
“a+“ - voog F-i lõppu
kui avada kirjutamiseks või lisamiseks F-i , mida ei ole, siis ta luuakse
püüd lugeda F-st, mida ei ole, on viga
mistahes vea korral tagastab fopen viida **NULL**
kui F on avatud, siis saab sinna kirjutada või sealt lugeda, näit.:
c = getc (fp) - loeb ‘c‘-sse järgmise sümboli fp-st
putc (c, fp) - kirjutab sümboli c fp-sse
fclose (fp) - suleb fp, soovitav teha kohe, kui võimalik (koormab puhvrit, kuhu ***putc*** kogub O)
kui P lõpetab töö normaalselt, siis suletakse kõik avatud F-d

F-de funktsioonid

open on analoogiline ***fopen***’iga, ta tagastab F’i i-node’i (int)
int fd;
fd = open(nimi, rwolek);
nimi - string
rwolek on: 0 - kirjutamiseks, 1 - lugemiseks, 2 - mõlemaks
püüd avada F’i, mida ei ole, on viga; vea korral tagastab ‘-1‘
uute F-de loomiseks ja vanade puhastamiseks - ***creat***
inode = creat(nimi, kasutusluba); 755
close katkestab sideme fd ja avatud F’i vahel
exit või ***main***’ist väljumine suleb kõik avatud F-d
unlink(nimi) - eemaldab ‘nimi‘ F-S-st
F-d - järjestikF-d
lseek(fd, offset, origin) - liigume offset võrra origin’i suhtes
järgnev tegevus toimub sellest punktist
offset on ***long,*** fd ja origin - ***int***, origin = 0(F’i algus), 1(jooksev koht), 2(F’i lõpp)
lseek(fd, 0L, 2) - F’i lõppu
lseek(fd, 0L, 0) - F’i algusesse

17.04.97 5.

protsessid

protsess on töötav P
mingil ajahetkel võib P-le vastata 0 või rohkem protsessi
protsess on objekt, mis on T spetsiaalses tabelis
tegelikult on oluline info protsessi kohta 2-s kohas:
• protsesside tabelis
• tarbija tabelis
1. on kogu aeg mälus ja seal on iga protsessi kohta 1 element, kus kajastub protsessi olek (aadress, suurus, identifikaatorid)
2. on iga protsessi kohta
protsesside juhtimiseks on komplekt süsteemseid P-e:

fork	- luua protsess	kahvel
exec	- käivitada F	
exit	- lõpetada protsess	
kill	- saata signaal protsessile	tapa ?
signal	- anda vastus signaalile	
wait	- oodata protsessi lõppu	

enamikes OS-des on võimalus panna P-d tööle üksteise järel
lihtsaim viis käivitada 1 P 2.-st on:

```
main()  {
        system("date");       }
```

kui ei kasutata standardkataloogi, siis tuleb minna veel “alla“

```
#include <stdio.h>
main()
{       execl("/bin/date", "date", NULL);       }
```

1. a - Fnimi F-S-s 2. a - Pnimi, s.t. Fnime viimane komponent
NULL - listi lõpp
execl asendab olemasoleva P-i uuega, käivitab selle ja lõpetab prots.
NB! **<u>ei toimu</u>** tagasipöördumist esialgsesse - puudub tagasiside
ainus erand - viga