UNIX FELHASZNÁLÓI ALAPISMERETEK

BEVEZETÉS

A Unix operációs rendszer egy multiuser, multitasking operációs rendszer. A Unix a C programozási nyelvvel együtt a 70-es években született az AT&T-nél. A Unix kb. 90%-át C nyelven írták, így nagy mértékben hordozhatóvá vált, azaz sok (különféle processzorral működő) számítógéptípusra könnyen telepíthető. Az AT&T mellett a Unix fejlődésében nagy szerepe volt a kaliforniai Berkeley Egyetemnek is. A Unix-nak két alapváltozata létezik: AT&T Unix és Berkeley Unix. Az egyes számítógépgyártó cégek a maguk Unix változatát ezek valamelyikéből alakították ki. A Unix-nak vannak ingyen hozzáférhető, ún. public domain verziói is. Ilyen pl. az IBM PC-ken futtatható LINUX is. A Digital Equipment Corporation (DEC) VAX ill. DECStation típusú gépein általában a DEC saját Unix változata, az Ultrix fut. A Sun Microsystems munkaállomásain illetve szerver gépein a Unix SUN-OS, vagy Solaris nevű változata hozzáférhető. A sokféle Unix változat mellett létezik egy ún. 'szabványos' Unix is, az Open Software Foundation által definiált Unix, az OSF-Unix.

A Unix sok előnye mellett (szinte minden géptípuson fut, ingyen változatai is vannak, flexibilis, sok elegáns rendszerprogramozási fogást támogat) egyik nagy hátránya, hogy nem egy tudatos, célirányos ipari fejlesztőmunka eredménye, hanem egy tágabb számítástechnikus közösség lassan 2 évtizedes munkája során alakult ki. Ezt mutatja az is, hogy a Unix nem kifejezetten felhasználóbarát, a számítástechnikában (Unix-ban) gyakorlatlan felhasználó nehezen tudja kezelni. (Nehezen megjegyezhetőek a Unix parancsai, sok egykarakteres parancs-sor kapcsolóval vezérelhető a működésük, a Unix help-rendszere a Unix-ra vonatkozó átfogó ismeretek nélkül alig használható.) Ezt a hátrányt úgy igyekeznek az egyes cégek feloldani, hogy a Unix fölé valamilyen könnyen kezelhető, ablakozós grafikus felhasználói felületet telepítenek (lásd pl. a SUN munkaállomásokon az OpenWindows rendszert). Ezek a grafikus felhasználói felületek többnyire a szabványosnak tekinthető X-Windows rendszeren alapulnak.

Jelen leírásban a DOS/Windows rendszerrekkel kapcsolatban már megszerzett felhasználói tudásra támaszkodva igyekszünk bemutatni a legalapvetőbb Unix felhasználói ismereteket. A Unix felhasználói ismeretekkel kapcsolatban nem törekszünk teljességre; csak a legfontosabb dolgokat közöljük, és azokat is csak olyan mélységben, hogy a kezdeti lépéseket a Unix-szal most ismerkedő olvasó önállóan is megtehesse. Ezt követően leírjuk, hogy miképp használható a BME Villamosmérnöki és Informatikai Karának számítóközpontjában (HSZK) telepített URAL2-es számítógép. Leírásunkat elsősorban az 1. éves műszaki informatika szakos hallgatóknak szánjuk a *Programozás alapjai* c. ill. *Számítógépek programozása labor* c. tantárgyak 2. féléves segédleteként.

ELSŐ LÉPÉSEK A UNIX-BAN

A Unix-ban az információk egysége - a *byte* mellett - a *file*. Ez azt jelenti, hogy a tényleges adathordozón található információk mellett minden egyéb "adatforrást" (pl. terminál billentyűzete) vagy "adatnyelőt" (pl. printer) formálisan file-ként kezel az operációs rendszer. Speciális file-ok az ún. *directory*-k, amelyek más file-ok adatait tartalmazzák. A legtöbb esetben a Unix-os directory fogalom megegyezik a DOS-ban megismert *könyvtár* fogalmával. A Unix-ban hierarchikus, fa jellegű directory struktúra van. A struktúra gyökere a *root* könyvtár, amelynek jele a / szimbólum (slash - "osztás" szimbólum). A továbbiakban csak a *könyvtár* jellegű directory-król szólunk, az egyes fizikai eszközökre (pl. terminálok, nyomtatók, munkaállomás-képernyők, mágnesszalag egységek) vonatkozó bejegyzésekről (amelyek mind a **/dev** directory-ban találhatók) nem szólunk.

A Unix minden egyes felhasználóhoz egy saját diszk-területet rendel. Ez - durván fogalmazva - a felhasználó ún. *login könyvtára* vagy *home directory*-ja. A Unix-ban a home directory nevét a teljes hozzáférési úttal együtt a **$HOME** szimbólum hordozza.

Ahhoz, hogy hozzáférjünk egy Unix operációs rendszert futtató géphez, rendelkeznünk kell a hozzáférési jogosultsággal (account). Ha rendelkezünk a jogosultsággal, szükséges még egy program, mellyel - a hálózaton keresztül - elérjük az adott gépet. Ez a program lehet a TELNET nevű program, amely egy terminál kapcsolatot biztosít nekünk. Újabban a hálózaton keresztül megnövekedtek a visszaélések, ezért a terminál kapcsolat biztosításához egy biztonságos csatornát - SSH - kell használnunk, amennyiben megköveteli az adott gép üzemeltetője (az URAL2 ilyen). Ezt a biztonságos terminál kapcsolatot bármely program biztosíthatja, mely tud SSH-val kommunikálni. Windows operációs rendszerre ilyen program a putty.exe, mely letölthető a "http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html" címről.
Tehát egy Unix-os gépre a

```
login:
Password:
```

kérdésekre adott helyes válaszokkal léphetünk be. Ez a mechanizmus védelmet jelent az illetéktelen felhasználók ellen, illetve lehetőség van a személyes, bizalmas jellegű adatállományok védelmére. A felhasználók beállíthatják, hogy egyes állományaikhoz kik férhetnek hozzá, és mit csinálhatnak az egyes file-jaikkal (olvashatják-e, írhatják-e, futtathatják-e). Egy

állomány tulajdonjoga, illetve elérhetősége szempontjából a felhasználók felosztása a következő. Létezik a *owner* (az a felhasználó, akié az adott file), a *group* (a tulajdonossal egy csoportban lévő felhasználók halmaza) és a *world* (az összes felhasználó halmaza).

Az egyes felhasználók tehát csoportokra vannak osztva és a csoporton belül is önálló azonosító számmal rendelkeznek. A csoportazonosító szám jele a GID (Group IDentifier), a felhasználói azonosító szám jele az UID (User IDentifier). A BME Villamosmérnöki és Informatikai karán a Unix operációs rendszert futtató és egységes hálózatba szervezett számítógépek felhasználói esetében a használható GID és UID értéktartományok egységes elvek szerint lettek meghatározva. Ezen elvek betartásával minden egyes felhasználónak a saját szervezeti egysége (tanszék, hallgatóknál a HSZK) ad egy konkrét UID és GID értéket. A kari számítógép-hálózat használata szempontjából e két szám tehát olyan szerepet tölt be, mint a *személyi szám.* A hallgatóknak a HSZK ad UID-et és GID-et, valamint a HSZK biztosít saját diszk-területet a URAL2-es számítógépen.

FILE-OK SPECIFIKÁCIÓJA

Az operációs rendszer a diszket hierarchikus könyvtárstruktúrába szervezi. A Unix-ban egy adott felhasználói file teljes specifikációja a következőképpen néz ki:

homedir/*subdir*/ ..... /*subsubdir*/*filename*

ahol

homedir	az adott felhasználó saját könyvtára
subdir	
....	egy adott alkönyvtár-sor elemei
subsubdir	
filename	a kérdéses file neve

A file-ok nevei (*filename*) betűkből, számokból és az operációs rendszer által egyéb célra nem használt speciális karakterekből állhatnak. **Fontos, hogy a Unix különbséget tesz a kis- és a nagybetűk között!** A Unix nem ismeri a file-név kiterjesztés fogalmát, de minden file-név tartalmazhat pontot (**.** karakter), így utánozhatjuk a DOS/Windows-ban megszokott file-név kiterjesztéseket. A Unix-ban fontos file-név konvenció az, hogy a C nyelvű forrásszövegeket tartalmazó állományok **.c**,illetve **.h** végződésűek, valamint az egyes tárgykódú modulokat tartalmazó állományok neveinek végződése **.o**. A Unix file-név konvenciókkal kapcsolatban fontos még, hogy a ponttal kezdődő nevű file-ok ún. rejtett állományok, azaz egy könyvtár tartalmának listázásakor alapértelmezésben *nem jelennek meg.*

HELP A UNIX PARANCSOKRÓL, AZ EGYES INSTALLÁLT PROGRAMOKRÓL

man *téma*

A Unix **man** parancsa szolgál arra, hogy egy adott *téma* kapcsán tájékoztató információhoz jussunk. Itt *téma* egy olyan Unix parancs, Unix fogalom, vagy az adott gépen man-helppel installált program neve kell legyen, amelyről információt szeretnénk kérni. A *téma* ismerete nélkül a **man** parancsot nem tudjuk használni! A **man** használatáról a **man man** paranccsal kaphatunk információt.

Fontosabb témák:

cd	change directory - könyvtár váltás
pwd	aktív könyvtár lekérdezése
ls	könyvtár tartalmának listázása
rm	file-ok törlése
mv	file-ok mozgatása/átnevezése
cp	file-ok másolása
mkdir	alkönyvtár készítése
cat	file tartalmának listázása a standard outputra
more	karakterfolyam kijelzése a képernyőnek megfelelő tagolásban
alias	szimbólumok definiálása/lekérdezése
set	környezeti változók definiálása/beállítása/lekérdezése
history	kiadott parancsok listája
who	bejelentkezett felhasználók listáját adja
ps	tájékoztatást ad a futó processzekről
kill	folyamatok (processzek) lelövése
grep	sztring-minta keresés file-okban

cc	C fordító
make	pl. több modulos C programok intelligens fordításához jó
vi	tetszőleges terminálon használható szövegszerkesztő
ftp,telnet	hálózati programok, saját help-jük van a *help* paranccsal
rlogin	átjelentkezés egy másik számítógépre

Példa a **man** parancs használatára - információt kérünk a **cp** parancsról (file-ok másolása):

```
ural2% man cp
```

A következőkben témakörök szerint röviden ismertetjük a legfontosabb Unix parancsokat. Tekintve, hogy egy-egy parancsnak számos kapcsolója lehet, javasoljuk, hogy mindenki alaposan tanulmányozza a **man**-nal nyerhető információkat, hogy az általunk közölt szűkös ismereteken túlmenően is használhassa az egyes parancsokat.

DIRECTORY-K

A directory-k speciális file-ok, amelyek egyéb file-ok (vagy perifériák) adatait tartalmazzák. A directory-k aldirectory-kat tartalmazhatnak, különböző feladatokhoz tartozó file-ok elkülönítésére.

Directory készítése

Unix:	*DOS:*
mkdir *dirname*	**md** *dirname*

Az aktuális directory-ban egy *dirname* nevű alkönyvtárat készít. Az alkönyvtárak egy *dirname* nevű file-ként látszanak az aktuális könyvtárban.

Directory váltás

Unix:	*DOS:*
cd *subdir*	**cd** *subdir*

Az aktuális könyvtár *subdir* nevű alkönyvtárára állítjuk át a aktuális könyvtárat. (Relatív specifikáció)

Unix:	*DOS:*
cd */dir*	**cd** *\dir*

A directory fában abszolút módon írjuk elő, hogy az aktuális könyvtár mire legyen állítva. Jelen példában a gyökér könyvtár *dir* nevű alkönyvtárába váltunk.

Unix:	*DOS:*
cd ..	**cd ..**

Visszalépés a directory fában a *root* irányába egy szinttel.

```
cd $HOME
cd ~
cd
```

A bejelentkezéskor érvényes könyvtárba (login könyvtár) vált. A Unix-ban a fent megadott 3 forma ekvivalens. Vigyázat: az argumentum nélküli **cd** parancs a Unix-ban nem az aktív könyvtár nevét adja meg (DOS), hanem a **$HOME**-ba vált. Észre vehetjük, hogy a ~ szimbólum (önmagában) az aktuális felhasználó login könyvtárát jelenti, míg a *~name* a *name* azonosítójú felhasználóét jelenti.

Az aktív könyvtár (default directory, vagy default path) lekérdezése

Unix:	*DOS:*
pwd	**cd**

A **pwd** parancs a *print working directory* angol szavak kezdőbetűiből áll.

Directory tartalmának megnézése

Unix:	*DOS:*
ls *mask*	**dir** *mask*

Az aktuális könyvtár *mask*-ra illeszkedő file-jait listázza ki a Unix. A *mask* elhagyása esetén az összes látható file-t listázza az **ls**. A Unix-ban a ***** *mask* az "összes file"-t jelenti.

Például:

$ ls *.h Az összes **.h**-ra végződő nevű file-t kilistázza:

```
xvars.h
```

Az **ls** parancsnak sok ún. kapcsolója (Unix terminológiával 'switch'-e) van. Ezek közül néhány fontosabbra mutatunk be példát:

$ ls -l E parancs hatására ún. hosszú (long) listát kapunk: az **ls** parancs megjeleníti a hozzáférési jogokat (lásd később), a file tulajdonosának az azonosítóját és a file-hoz való hozzáférés dátumát is.

Például egy könyvtárlista az **l** kapcsolóval:

```
total 6
-rw-r--r--  1 s1234abc     1239 Feb 25  1992 head.txt
-rwxr-x---  1 s1234abc     2288 Sep 29 10:51 joc2.txt
drwxr-xr-x  2 s1234abc      512 Dec  3  1992 termanal
drwxr-xr-x  2 s1234abc     1024 May 14  1993 tex
-rw-r--r--  1 s1234abc     5865 Mar  1  1993 xvars.h
drwxr-xr-x  3 s1234abc      512 Sep 28 15:59 xview
```

$ ls -a A **-a** kapcsoló hatására (all) minden file (így a ponttal kezdődő nevűek is!) megjelenik a könyvtárlistában.

Az előbbi könyvtárlista ekkor így néz ki:

```
.
..
.Xauthority
.Xdefaults
.cshrc
.desksetdefaults
.login
.logout
.openwin-init
.xinitrc
head.txt
joc2.txt
termanal
tex
xvars.h
xview
```

A **.** a DOS-hoz hasonlóan az aktuális könyvtárat jelenti, a **..** pedig ennek "szülőjét". A többi **.** kezdetű fájl különböző, rendszerhez kapcsolódó programok munkafájljai.

Egyszerre több kapcsolót is megadhatunk; ekkor ezek logikai ÉS kapcsolatát képezve kapjuk meg a file-ok listáját. Például az **ls -al** parancs hatására az összes file-t a hozzáférési jog, méret- és dátuminformációval együtt listázza a Unix:

```
total 16
drwxr-xr-x 12 s1234abc     1024 Jan  8 11:59 .
drwxr-xr-x 10 root          512 Oct  5 17:26 ..
-rw-------  1 s1234abc      833 Nov  8 15:28 .Xauthority
-rw-r--r--  1 s1234abc      361 Sep 21 17:31 .Xdefaults
-rwxr-xr-x  1 s1234abc     2681 Apr 23  1993 .cshrc
-rwxr-xr-x  1 s1234abc      784 Sep  1 19:12 .desksetdefaults
-rwxr-xr-x  1 s1234abc     1807 Apr 23  1993 .login
-rw-r--r--  1 s1234abc       61 Aug 28  1992 .logout
-rwxr-xr-x  1 s1234abc      365 Sep 21 17:31 .openwin-init
-rw-r--r--  1 s1234abc      766 Feb 17  1992 .xinitrc
```

```
-rw-r--r--  1 s1234abc     1239 Feb 25  1992 head.txt
-rwxr-x---  1 s1234abc     2288 Sep 29 10:51 joc2.txt
drwxr-xr-x  2 s1234abc      512 Dec  3  1992 termanal
drwxr-xr-x  2 s1234abc     1024 May 14  1993 tex
-rw-r--r--  1 s1234abc     5865 Mar  1  1993 xvars.h
drwxr-xr-x  3 s1234abc      512 Sep 28 15:59 xview
```

FILE-KEZELÉS

File-ok másolása

Unix:	*DOS:*
$ cp *sour dest*	**copy** *sour dest*

Ez a parancs a *sour* specifikációval megadott file-t a *dest* specifikáció szerinti file-ba másolja.
Például:

$ cp ../some.txt . A **../some.txt** file-t az aktív könyvtárba másoljuk.

A **cp** parancs használatának részleteit a **man cp** paranccsal kaphatjuk meg, angol nyelven.

File-ok átnevezése

Unix:	*DOS:*
mv *old new*	**rename** *old new*
	DOS 6.0-tól felfelé:
	move *old new*

Ez a parancs lényegében úgy működik, mint a DOS 6.x-beli megfelelője: az *old* nevű file-t *new*-ba helyezi át. Mind az *old,* min a *new* PATH specifikációt is tartalmazhat. (A **rename** csak részben hasonlít az **mv**-re, ill. a **move**-ra.)

Például:
$ mv ../something . Az aktív könyvtár "szülőjéből" a **something** file-t az aktív könyvtárba helyezzük át.

File-ok törlése

Unix:	*DOS:*
rm *mask*	**del** *mask*

A DOS **del** parancsához hasonlóan működik. A *mask* paraméter megadása kötelező. A *mask* egy file-specifikáció kell legyen, amelyben a szokásos *joker* (vagy *wildcard*) karakterek is megadhatók maszk gyanánt (lásd az **ls**-t). Pl. a ***** tetszőleges részsztringet helyettesíthet. Tartózkodjunk az **rm *** parancs kiadásától: ez mindent kitöröl az aktív könyvtárból.

Az **rm** parancsnál célszerű egyes kapcsolók (command line switches) használata. Ilyen például a **-i**

rm *mask* **-i**

hatására az összes, a *mask*-ra illeszkedő file törlésére vonatkozólag megerősítést kér az operációs rendszer. Az aktív könyvtártól lefelé minden állomány törlése (alkönyvtárakban is) a következőképpen lehetséges:

rm -Ri

Az **rm -R *** paranccsal nagyon körültekintően bánjunk! Az egyszer már kitörölt állományok helyreállítására nincs mód! A fenti példában a biztonság kedvéért a **-i** (/**confirm**) kapcsolót is megadtuk.

Célszerű, ha a könnyen reprodukálható állományaink közül a szükségteleneket rendszeresen töröljük. Így például egy C programozási feladat befejezése után javasoljuk a C forrásállományokból generált **.o** file-ok (object modulok) törlését, ugyanis ezek egy programfordítási paranccsal (lásd később) újra generálhatók:

Unix:	*DOS:*
rm *.o	**del *.obj**

ASCII SZÖVEGES FILE-OK TARTALMÁNAK MEGNÉZÉSE

Unix:	*DOS:*
cat *filespec*	**type** *filespec*

A DOS **type** parancsához hasonlóan működik: a *filespec*-cel adott ASCII szöveges állomány tartalmát megjeleníti a standard outputon (ami legtöbbször a képernyő). Hosszabb file-ok megnézésére kombináljuk a **cat**-ot a **more** programmal:

Unix:	*DOS:*
cat *filespec*| **more**	**type** *filespec* **| more**
more *filespec*	

Ennek hatására a *filespec* által megadott file-t képernyőnként listázza az operációs rendszer. A listázás egyébként CTRL-S-sel felfüggeszthető, illetve CTRL-Q-val folytatható. Ez általában is igaz minden output műveletre (lásd software-handshake). CTRL-C-vel megszakíthatunk egy futó programot, így a **cat**-ot is. A | (ún. *pipe*) szimbólum jelentéséről később szólunk. A sima **cat** *filespec* parancs egyszerűen csak a standard outputra listázza a megadott file-t.

KIJELENTKEZÉS A RENDSZERBŐL

A rendszerből való kijelentkezés azt jelenti, hogy befejezzük a parancsértelmező futtatását, valamint felszabadítjuk azt az I/O csatornát (terminált), amelyen keresztül be voltunk jelentkezve. Az ehhez szükséges parancs:

logout

A Unix-ban a parancsértelmező futása akkor is megszakad, ha a standard inputról 'állományvége' jel érkezik. Unix-ban ez a jelzés CTRL-D megnyomásával adható. Elképzelhető, hogy a bejelentkezés után valamilyen módon a parancsértelmező egy újabb példányát indítottuk el. Ekkor az alprocesszként futó parancsértelmező programpéldányt a következőképpen hagyhatjuk el:

exit

EGYÉB PARANCSOK

Az interaktív felhasználók listájának lekérése

who

E paranccsal megnézhetjük, hogy mely felhasználók vannak bejelentkezve, és hogy mely terminálokon dolgoznak. Ha arra vagyunk kíváncsiak, hogy a Unix promptnál éppen ki van bejelentkezve, adjuk ki a **who am i** (magyarul: *ki vagyok én*?) parancsot! Ha nagyon sok felhasználó van bejelentkezve, kombináljuk a **who**-t a **more**-ral:

$ who | more	A teljes felhasználói lista képernyőkre tagolva jelenik meg.

Ha egy adott nevű felhasználót keresünk, kombináljuk a **who** parancsot a **grep** programmal! Pl. ha azt szeretnénk megtudni, hogy az **s1234abc** felhasználó be van-e jelentkezve, akkor adjuk ki az alábbi parancsot:

$ who | grep s1234abc	Ha semmi nem íródik ki a képernyőre, akkor a keresett felhasználó nincs bejelentkezve.

Sztring keresése file-okban

grep *string mask*

A *mask* által megadott file-okban *string*-et keresi. Ha a *mask*-ot nem adjuk meg, akkor a keresést a standard inputon hajtja végre a **grep**. Ez lehetővé teszi azt, hogy I/O átirányítás esetében is használjuk a **grep**-et. Az I/O átirányításról később még szólunk.

Jelszó (password) átállítása

passwd

Ezzel állíthatjuk be a password-öt az általunk adandó új jelszóra. Így biztosíthatjuk, hogy személyes adatállományaihoz illetéktelen módon ne lehessen hozzáférni.

A parancs kiadása után a **New password** kérdésre adjuk meg az új jelszót - ami 6 karakternél hosszabb kell legyen és tartalmaznia kell egy számjegyet is -, majd a **Verification** kérdésre adott válasszal erősítsük meg - az új jelszó másodszori begépelésével - a változtatást. Az új jelszót csak akkor fogadja el a számítógép, ha a **Verification**-re adott válasz megegyezik a **New password**-ként megadott jelszóval.

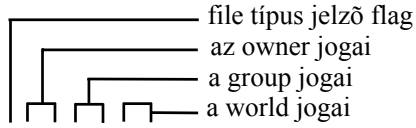
TOVÁBBI ISMERETEK

File-ok típusai, hozzáférési jogok

A Unix-ban egy diszk file vagy adatállomány, vagy futtatható állomány, vagy egy directory (alkönyvtár). Egy file típusának jellegét a rendszer a file-t tartalmazó directory-ban nyilvántartja. Az egyes állományok típusára vonatkozó információt az **ls -l** paranccsal kaphatjuk meg. Az így nyert a könyvtárlista első mezőjében, file-típust és az ún. hozzáférési jogokat különböző karakterek jelzik. Ezek közül a legfontosabbak:

d file-típus jelző: az állomány egy alkönyvtár (directory)
x az állomány futtatható
r az állomány olvasható
w az állomány írható
- az adott felhasználói csoportra nézve nincs jog engedélyezve, illetve file-típus jelző nincs beállítva

Attól függően, hogy a fenti karakterek mely pozíciókon állnak, megtudhatjuk, az egyes hozzáférési jogok mely felhasználói csoportok számára vannak engedélyezve. A felhasználók csoportosítása a Unix-ban: tulajdonos (owner), a csoport (group) és mindenki más (world).




```
drwxr-xr-x 12 s1234abc     1024 Jan  8 11:59 .
drwxr-xr-x 10 root          512 Oct  5 17:26 ..
-rw-------  1 s1234abc      833 Nov  8 15:28 .Xauthority
-rw-r--r--  1 s1234abc      361 Sep 21 17:31 .Xdefaults
-rwxr-xr-x  1 s1234abc     2681 Apr 23  1993 .cshrc
-rwxr-xr-x  1 s1234abc     1807 Apr 23  1993 .login
drwxr-xr-x  2 s1234abc     1024 May 14  1993 tex
-rw-r--r--  1 s1234abc     5865 Mar  1  1993 xvars.h
drwxr-xr-x  3 s1234abc      512 Sep 28 15:59 xview
```

Hozzáférési jogok állítása

chmod *jog_maszk file_maszk*

A **chmod** esetében a *jog_maszk* egy 3 jegyű oktális szám. Minden egyes számjegy egy-egy felhasználói csoportra vonatkozik (owner, group, world). Minden egyes bit (3 bit) egy-egy jog meglétét vagy hiányát jelzi.

Például minden file-ra minden jogot mindenkinek a **chmod 777 *** paranccsal adhatunk meg.

A külvilág hozzáférést a **chmod 770 *** paranccsal tilthatjuk le.

A **chmod**-ra vonatkozó részleteket a **man chmod** parancs segítségével angol nyelvű help formájában megkaphatjuk.

Parancs-file-ok és futtatásuk

Egy ASCII állományt, amennyiben az a parancsértelmező burok parancs- nyelvén írt szöveget tartalmaz, valamint könyvtárbejegyzésében a típusa **x** (executable, azaz futtatható állomány), nevének begépelése által végrehajthatunk. Az ilyen állományokat *shell script*-nek szokták nevezni. Futtatásuk hasonlít a DOS-ban a .BAT file-ok futtatásához.

A legelterjedtebb parancsértelmező burok a **csh**, illetve az **sh**. Minden shell scriptben olyan parancsokat hajthatunk végre, illetve olyan utasításokat írhatunk le, amilyeneket a kérdéses parancsértelmező burok (command shell) elfogad. Jellemző, hogy egy script-ben Unix programokat vagy további script file-okat futtatunk le, vagy szimbólumokat, környezeti változókat definiálhatunk, ciklusokat szervezhetünk, stb. A **csh** parancsértelmező onnan kapta a nevét, hogy parancsnyelve nagyon hasonlít a C programozási nyelvhez.

A **set** parancs szolgál arra, hogy bizonyos környezeti változóknak értéket adjunk. Fontos környezeti változók:

PATH - hozzáférési út programok futtatásához
TERM - terminál típusa (pl. VT100)
DISPLAY - X-Windows Display, ha az egy távoli gépen van.
HOME - a login-könyvtár neve

Példák a **set** parancs használatára:

`set`	kiírja az összes környezeti változó értékét
`set TERM=VT100`	a terminál típusát VT100-ra állítja
`set DISPLAY=alex:0`	az X-Windows kijelzőt az **alex** nevű host képernyőjére állítja be.

Egy ily módon beállított változó értékére **$***változó* alakú kifejezéssel hivatkozhatunk. Például a **$HOME** kifejezés a HOME változó (login könyvtár neve) értékét adja, a **$PATH** a PATH változó (file hozzáférési út) értékét adja. Egy kifejezés értékét az **echo** paranccsal jeleníthetjük meg a standard outputon. Például: **echo $PATH**.

Helyettesítő szimbólumok definiálása

A Unix-ban az **alias** parancs segítségével egy sztringet helyettesítő szimbólumot definiálhatunk. A parancs általános formája:

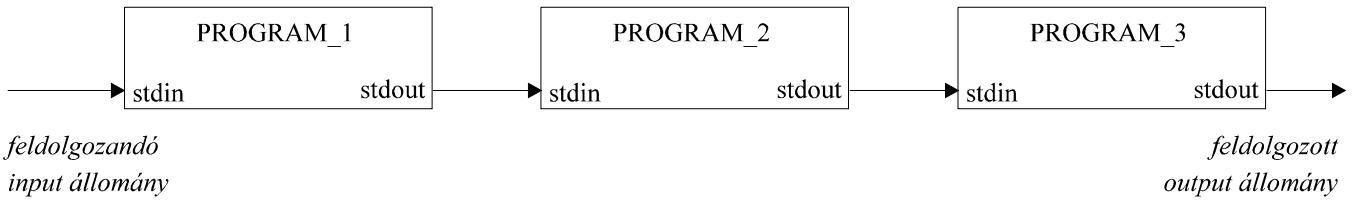
alias *mire mit*

Ez a parancs a *mit* sztringet a *mire* helyettesítő szimbólumhoz rendeli.

Érdemes a saját környezetet kialakító **set** és **alias** parancsokat a **.login** vagy **.profile** állományunk végéhez hozzáfűzni - az alkalmazott login shelltől függően -,. így a más operációs rendszerekben megszokott parancsokhoz hasonlóakat hozhatunk létre, a saját ízlésünknek megfelelő módon. Például: **alias dir ls**. Az argumentum nélkül kiadott **alias** parancs a létező szimbólum definicókat listázza ki.

I/O átirányítás a Unix-ban

A Unix operációs rendszer egyik nagy erőssége a folyamatok (az operációs rendszer által futtatott programok) szabványos ki- és bemeneti állományainak átirányíthatósága, és az ennek révén több, különböző programból felépített adatfeldolgozó rendszerek, ún. adatfolyamok létre- hozásának lehetősége. Egy adatfolyamot úgy képzelhetünk el, mint egy csővezetéket (*pipe*), amelynek az elején (standard input) betöltünk valamilyen folyadékot (adat). A csőben különböző szűrők lehetnek (filter programok, vagy egyszerűen csak filterek), amelyek megváltoztatják a csőben áramló közeg (adatok) jellegét. A csőből egy megváltoztatott jellegű közeg (feldolgozott adatok) folyik ki a kimeneti csapon (standard output). Egy ábra segítségével így jellemezhető egy adatfolyam:



A Unix adatfeldolgozó pipe-ba a < szimbólummal "tölthetünk be" adatokat: így irányítjuk át az első feldolgozó program standard inputját egy file-ba. A > szimbólummal a standard outputon "kiömlő" adatokat egy kimeneti állományba "tölthetjük". Példaként tekintsük a **sort** programot, amely standard inputról érkező állomány sorait abc sorrendbe rendezve kiírja a standard outputra! Ha ezzel egy rendezetlen ASCII szövegfile-t szeretnénk sorba rendezni (pl. tankör névsor rendezése), akkor ezt az alábbi paranccsal tehetjük meg:

`(sort <` *`rendezetlen`***`) >`** *`rendezett`*

Láthatjuk, hogy a parancsértelmező számára a **()** zárójelekkel tagoltuk ezen egyszerű adatfolyamunk leírását.

Ha több programot akarunk egy folyamba rendezni, akkor erre az ún pipe szimbólumot, a | jelet használjuk. Például:

`who | grep s5 > info1users`

A **who** program előállítja a standard outputon az éppen bejelentkezett felhasználók listáját. Ezt az adatfolyamot a **grep** program szabványos bemenetére irányítjuk a | szimbólum segítségével. A **grep** számára az **s5** keresési kulcsot adtuk meg első parancssor paraméterként. Tekintve, hogy nincs második paramétere, a **grep** tudja, hogy a standard inputján hajtsa végre a keresést. A

grep a szabványos kimeneten listázza azon sorokat, amelyekben a keresési kulcsot megtalálta, ugyanakkor a fenti parancs-sorban a végső kimenetet az **info1users** állományba irányítjuk át. Több program összefűzésére példa:

```
(who | grep s5) | more
```

A fenti parancs ugyanazt csinálja, mint az előző, de a **grep** szabványos kimenetén jelentkező adatfolyamot a **more** program segítségével a terminálon, képernyőnyi méretű részekre tagolva jeleníttetjük meg.

SAJÁT PROGRAMOK ÍRÁSA C NYELVEN: FORDÍTÁS, LINKELÉS, FUTTATÁS

Programjainkat a Unixon valamilyen szövegszerkesztő programmal (például **joe**) gépelhetjük be (vagy a hálózatba kapcsolt más gépekről a URAL2-re másolhatjuk). A forrás állományok elkészülte után azokat le kell fordítani a megfelelő fordító programmal, az így keletkező tárgykódú állományokból futtatható programot kell szerkesztenünk. Az így elkészült programot a Unix operációs rendszerből futtathatjuk. A Unix C fordítója (**cc**) alapértelmezésben össze is szerkeszti a futtatható programot, ha a fordítás hibátlan volt. A futtatható program neve **a.out** lesz, ha nem specifikálunk mást.

C fordító

Unix:	*DOS:*
cc *name*.**c**	**bc** *name*.**c** (Interaktív Borland C++)

A Unix **cc** fordítóprogram **.c**-re végződő nevű C forrásállományokat fordít le. A lefordított tárgykódú programmodul neve megegyezik az eredeti forrásmodul nevével, de **.o**-ra fog végződni. Ha lefordított forrásállomány egy teljes C program, és semmilyen parancs-sor kapcsolót nem adtunk meg a **cc**-nek, akkor automatikusan futtatható programot szerkeszt a **cc** (úgy, hogy behívja az **ld** programot) és a futtatható program neve **a.out** lesz. A **cc** program fontosabb parancs-sor kapcsolói:

-o a kész, futtatható program nevét adhatjuk meg vele
-c csak fordítást kérünk, azaz a **.c** állomány(ok)ból csak a **.o** állomány(oka)t kell elkészíteni
-l a szerkesztő program számára egyes futási idejű könyvtárakat specifikálhatunk.

Példák:

$ cc elso.c -oelso	Az **elso.c** állományban lévő teljes program fordítását és futtathatóvá szerkesztését kérjük. A futtatható program neve **elso** legyen.
$ cc masodik.c -c	A **masodik.c** forrásból csak a **masodik.o** tárgykódú modult kell elkészíteni; szerkesztés nem kell.
$ cc mat.c -omat -lm	A **mat.c** forrásállomány egy olyan teljes C program, amely matematikai függvényeket is használ (a **math.h**-ból), ezért a futtatható programhoz hozzá kell szerkeszteni a matematikai könyvtárat is (**-lm** kapcsoló). A futtatható program neve **mat** legyen.

A **cc** program parancs-sorában több állomány nevét is megadhatjuk. Az egyes **.c** állományokat sorra veszi, és egyenként lefordítja azokat. Ha **.o**-ra végződő file-nevet adunk meg, akkor ezt a nevet egyszerűen csak tovább adja a **cc** az **ld** szerkesztő programnak, így a korábban már elkészített tárgykódú moduljainkat is hozzászerkeszthetjük az aktuálisan lefordított C programmodulhoz.

Például:

$ cc x1.c, x2.c, y1.o -oxx	Az **x1.c** és **x2.c** állományok fordítását kérjük. A kész program neve **xx** legyen, és ezt az **x1.o**, **x2.o** és **y1.o** object modulokból kell összeszerkeszteni.

Több forrásmodulból álló program készítésére azonban célszerű a Unix **make** segédprogramját használnunk. A **make** program számára egy **Makefile** nevű állományban kell a teendőket - pl. a **cc** program hívását a megfelelő paraméterekkel - leírni. (Részleteket lásd a **man make** paranccsal).

Programfuttatás

A Unix-ban minden futtathatónak jelzett file-t (x-szel jelölve az **ls -l** paranccsal kapott listában) programként futtathatunk úgy, hogy a nevét begépeljük, mintha az egy Unix parancs lenne. (Ugyanúgy járunk tehát el, mint a DOS-ban.) Ha a kérdéses file

egy ASCII szöveges állomány, akkor azt a parancs-értelmező burok (command shell) úgy tekinti, mintha a parancs-értelmező saját nyelvén írt parancs file lenne (lásd: DOS .BAT file-ok). Az "igazi" futtatható bináris állományokat (pl. lefordított és összeszerkesztett C programokat) a command shell átadja az operációs rendszer más részének futtatásra.
Fontos, hogy csak a PATH környezeti változóban megadott direktorykban keresi a futtahtó állományt a rendszer, tehát ha az aktuális könyvtárból akarunk futtatni, akkor azt vagy hozzá kell illesztenünk a PATH változóhoz, vagy ./myprog begépelésével kell indítanunk.

Példa:

`$ ./xx` Lefuttatja a fenti **cc** paranccsal készített **xx** programot az aktuális könyvtárból.

A Unix-ban igen egyszerűen futtathatunk egy programot háttér folyamatként (ún. background process-ként). Ügyeljünk azonban arra, hogy egy háttérben futtatott program standard inputja és standard outputja a terminálról egy-egy file-ba, vagy adatfolyamba legyen átirányítva! (Ellenkező esetben összekeveredhet a parancsértelmező és a háttérben futó program adatforgalma.)

Példák:

`$ xx &` Az **xx** program a háttérben fog futni.

`$ yy > out.txt &` Az **yy** program a háttérben fog futni úgy, hogy a standard outputjára írt szöveg az **out.txt** file-ba kerül.

Ha egy programot a háttérben indítottunk el, akkor az indítás után mindig kiírja a Unix a képernyőre a folyamat azonosító számot, a *pid*-et (process identifier-t). Ezt jegyezzük meg, mert enélkül pl. a végtelen ciklusba keveredett programot nem tudjuk lelőni.

Futó programok lelövése

Egy nem háttérben futó programot a következőképpen lőhetjük le:

<CTRL-C>

Ha egy programunk a háttérben fut, akkor a folyamat azonosítójának ismeretében tudjuk csak lelőni azt:

`kill -9` *`pid`*

Ha nem tudjuk a *pid*-et, akkor a futó folyamatokról tájékoztatást kell kérnünk:

`ps -al`

Ekkor megjelenik egy lista, melyben találunk egy PID feliratú oszlopot, valamint láthatjuk a folyamat nevét is, ami a Unix-ban azzal a paranccsal egyezik meg, amellyel a folyamatot elindítottuk. Válasszuk ki a lelövendő folyamathoz tartozó *pid*-et, majd ennek ismeretében állítsuk le a kívánt programot. Vigyázat! Nehogy a saját command shell-ünk *pid*-jét adjuk meg paraméterként!