Számítógép architektúrák

A burok

Általános INFORMATIKAI Tanszék

A mai témáink

- **A Unix sh (bash) burok**
- **Fontos Unix burok parancsok nevei**
- **A parancs, cső, lista fogalom**
- **Adatfolyam átirányítás**
- **Fájlnév behelyettesítés**
- **Metakarakterek semlegesítése**
- **Burok adatszerkezetek, változó definiálás, változó behelyettesítés, érvényesség, környezet**

Általános INFORMATIKAI Tanszék

A Unix

- **Nagyon elterjedt, több processzes, időosztásos, általános célú OS**
- **Sok könyv, ismertető (hivatkozások az ütemtervben, jegyzetben)**
- **Parancsnyelvi kapcsolattartói (burkok):**
  - **Bourne shell sh $**
  - **Korn shell ksh $**
  - **C shell csh %**
  - **TC shell tcsh %**
  - **Bash bash $ (Bourne Again Shell)**

Általános INFORMATIKAI Tanszék

Unix filozófia

- **Mások munkájára építs!**
- **Ezt segítik a következő koncepciók:**
  - **processz általában az stdin-ről olvas, stdout-ra (stderr-re) ír; a parancsok is!**
  - **A standard adatfolyamok átirányíthatók.**
  - **Csővezeték képezhető. Szűrők használhatók.**
  - **Sok segédprogram (utility) létezik**
  - **Sok *daemon* processz szolgáltat.**

```
$  cat  txt
$  cat  >txt
$  grep  vmi  <txt | wc
```


Honnan tanulhatunk?

- **Könyvekből, az *Orlando* iskolából, *insight*-ból az Irix-ben, *info*-ból az AIX-ben stb.**
- **Az on-line kézikönyvből:**

> man [-opc] [section] lap

Tanácsok:

- **Tudni kell angolul olvasni!**
- **Fontos parancsok nevét pontosan tudni!**
- **“Parancs-kártya“ a fontos parancsok nevével!**

Parancsok: editorok

- **ed** sororientált
- **vi (vim)** képernyő orientált
- **mcedit** képernyő orientált
- **pico, nano egyszerű, sok helyen** (vt100 kell)
- **joe**
- **stb.**

Parancsok: kiírók

- **cat** összefűz, stdout-ra
- **pr** nyomtat, stdout-ra
- **head** fájl első sorait, stdout-ra
- **tail** fájl utolsó sorait
- **more** lapokra tördelő szűrő
- **od** oktális ömlesztés (dump)

Parancsok: jegyzékekkel kapcsolatban

- **ls** jegyzék tartalom lista (dir helyett)
- **mkdir** jegyzék készítés
- **rmdir** jegyzék törlés
- **cd** munkajegyzék váltás
- **pwd** munkajegyzék lekérdezés
- **chmod** fájl védelmi maszk váltás **(Nemcsak jegyzékre)**
- **chown** fájl tulajdonos váltás **(Nemcsak jegyzékre)**
- **file** fájl típus lekérdezés **(Nemcsak jegyzékre)**

Parancsok: másolások, mozgatások

- **cp** copy, másolás
- **mv** move, mozgatás (rename helyett is!)
- **ln (link)** "linkel"
- **rm (unlink)** "linket" töröl, remove: file törlés
- **find** keres fájlt egy fán és csinál is valamit (bonyolult, de nagyon hasznos!)

Parancsok: állapotlekérdezések

- **ps** — **processzek listája**
- **file, ls, pwd** — **ld. fönn**
- **date** — **dátum, idő lekérdezés**
- **who, w, rwho, rusers** — **ki van bejelentkezve?**
- **rup** — **mely rendszerek élnek?**
- **top** — **erőforrás használat csúcsok**
- **osview, vmstat** — **erőforrás használat**
- **last** — **utolsó bejelentkezések**
- **uptime** — **mióta fut a rendszer?**

Parancsok: állapotlekérdezések$_2$

- **finger** — **ki kicsoda?**
- **passwd, yppasswd** — **jelszóállítás**
- **chsh, chfn, ypchpass** — **név, induló burok stb. beállítás**
- **ypcat** — **NIS (yellow pages) adatbázis lekérdezés**
- **ldapsearch** — **LDAP adatbázis lekérdezés**
- **xhost** — **X11 munka engedélyezése**
- **set** — **környezet (environment) lekérdezése**
- **du, df, quota** — **diszk, fájl használat**

Parancsok: processz indítás, vezérlés

- **sh, csh, ksh, tcsh, bash** — **shell indítás**
- **exec** — **processz indítás**
- **kill** — **processz "megölése", szignálküldés**
- **sleep** — **processz altatása**
- **wait** — **processz várakoztatás**
- **at** — **processz indítása egy adott időpontban**
- **nohup** — **kilépéskor ne ölje meg**
- **test** — **kifejezés tesztelése**

Parancsok: processz indítás, vezérlés$_2$

- **expr** kifejezés kiértékeltetése
- **if, case, for, do while** vezérlő szerkezetek
- **break, continue** vezérlő szerkezetek
- **echo** argumentumai visszaírása (meglepően hasznos valami)
- **mplayer (video lejátszó)**
- **xmms, aumix (audio lejátszás)**

Parancsok: kommunikáció

- **ssh, telnet, rlogin, rsh** kapcsolatlétesítés
- **rwho, rusers, finger** lásd állapotlekérdezések
- **write** üzenet konzolokra
- **talk, xtalk** interaktív "beszélgetés"
- **mail, mutt, pine, mozilla-thunderbird** e-mail
- **ftp, scp** fájl átvitel
- **lynx, w3m, firefox, netscape** WWW böngésző (kliens)

Parancsok: hasznos szűrők

- **grep** mintakereső
- **awk, nawk** mintakereső feldolgozó
- **wc** sor, szó, karakterszámláló
- **sed** áradatszerkesztő
- **cut** mezőkivágó
- **tail, head, more** egyfajta kiírók
- **sort** rendező

Parancsok: tanuljunk

- **man** **on-line kézikönyv lap lekérdezés**
- **apropos** **kézikönyvben kulcsszó (ha van)**
- **whereis** **hol van egy parancs**
- **whatis** **man lap leírás**
- **xman** **X11-es kézikönyv (grafikus)**

A burok (sh) kifejezés kettős értelme

- **A burok egyrészt parancsértelmező processz,**
- **másrészt egy programnyelv.**
- **Mint processz: a /bin/sh betölthető, futtatható program egy futási példánya**
- **Mint programnyelv: egy szövegfájl, ami parancsokat (esetleg beleértett input sorokat) tartalmaz, és odaadható a burok processznek, hogy futtassa …**

A burok processz

- **Önálló entitás, azonosítója a pid (process identification number)**
- **A /bin/sh (vagy /bin/bash) program fut benne**
- **Van 3 nyitott adatfolyama**
  - **A 0 leírójú stdin (szabványos bemenet), ahonnan a parancsokat, csöveket, parancslistákat olvassa.**
  - **Az 1 leírójú stdout (szabványos kimenet), ahová az eredményeit írja.**
  - **A 2 leírójú stderr (szabványos hibakimenet), ahová a hibaüzeneteit írja.**
- **A nyitott adatfolyamok „szokásos módon” eszközökhöz vannak kapcsolva**

A burok processz működése

- **Az stdout csatornájára kiírja a készenlét jelet (prompt), jelezve, hogy parancsot, csövet, parancslistát vár**
- **Az stdin csatornáján parancsot, csövet, parancslistát olvas be,**
  - **Azt elemzi, értelmezi,**
  - **átalakítja, majd végrehajtja, vagy végrehajtatja.**
- **A végrehajtás eredményét az stdout, ill. stderr csatornára írja, végül visszatérési értéket produkál.**

A visszatérési érték

- **Lehet normális (0),**
- **lehet nem normális (nem 0), ennek oka többféle**
  - **valami hiba van,**
  - **nincs hiba, de szemantikailag van gond.**

  (Pl. grep szűrő nem talál minta-egyezést, vagy test parancs tesztelése nem igaz.)
- **A visszatérési értéket a programvezérlésben használhatjuk majd.**

A parancs fogalma

- **Fehér karakterekkel határolt szavak sora**
  - **első szó a parancs neve,**
  - **többi szó az argumentumok.**
- **Az sh beolvassa, értelmezi, átalakítja, végrehajtja**
  - **saját maga (belső p.),**
  - **gyermek processzben (külső p.)**

Mindkét esetben van visszatérési értéke!

Vannak szabványos adatfolyamok!

Egy példa parancsra

> find . -name a.c -print

ahol a szavak számozása

0 1 2 3 4

Azaz a fenti parancs 5 szóból áll. Vegyük észre, hogy a burok promptja nem része a parancsnak! Inputot nem kíván, outputja (esetleges hibaüzenetei is) a képernyőre megy. Vajon mi a visszatérési értéke? És ez?

> find . -name a.c -print >myfile.txt

A csővezeték fogalma

- **A csővezeték (pipe) parancsok sora | operátorral összekötve:**
- **parancsbal | parancsjobb**
- **Szemantikája: végrehajtódik a parancsbal, szabványos kimenete egy csőbe képződik, majd végrehajtódik a parancsjobb, aminek szabványos bemenete erre a csőre képződik.**
- **A cső visszatérési értéke: a parancsjobb visszatérési értéke.**
- **A parancs degenerált cső. Példa:**

> ypcat passwd | grep kovacs

A parancslista

- **Csővezetékek sora listaoperátorral összekötve:**

csőbal op csőjobb

Listaoperátorok:

&& || # magasabb precedencia, de alacsonyabb mint a |

& ; \n # alacsonyabb precedencia

A szemantika:

; \n soros végrehajtása a csöveknek

& aszinkron végrehajtás (csőbal háttérben)

&& folytatja a listát, ha csőbal normális visszatérésű

|| folytatja a listát, ha a csőbal nem normál visszatérésű

Parancslisták

- **A lista visszatérési értéke az utolsó cső visszatérési értéke.**
- **Háttérben futó cső visszatérési értéke különlegesen kezelhető.**
- **A cső degenerált lista (ahol ezentúl listát írunk, írhatunk csövet, sőt parancsot is!)**
- **A && és || operátoros listáknál először láthatjuk a visszatérési érték értelmét! Valóban a vezérlés menetét befolyásoljuk!**

Példák

> cd ide && rm junk # csak akkor töröl, ha ...

> ls ide || cp valami ide # ha nincs ide, készíti

Nézzük, magyarázzuk ezt!

> (mv a tmp && mv b a) && mv tmp b

Példák

- **„Háttérben” futtatunk**

> myprog 1 2 &
125
>

- **Mi a különbség?**

```
> echo valami echo valami
valami echo valami
```

```
> echo valami; echo valami
valami
valami
```

Az adatfolyamok átirányítása

- **Mielőtt a lista/parancs végrehajtódik, az sh nézi, van-e átirányító operátor > >> < a szavakban (szavak előtt). (A << különleges!)**
- **Ha ilyeneket talál, szeparált processz(eke)t készít, azokban az adatfolyamokat fájlokba(ból) képzik le, majd abban hajtják végre a listát/parancsot. (Csőnél is szeparált processz!)**
- **A szeparált processz(ek)nek átadja a "maradék" argumentumokat.**

Az átirányító operátorok

< file # file legyen az stdin
> file # file legyen az stdout, rewrite
>> file # file legyen az stdout, append
<<[-]eddig # here document, beágyazott input

- **Példa:**

> mypr < innen > ide elso masodik
0 1 2

> exec >outfile 2>errorfile # szkriptben …

Fájlnév kifejtés (behelyettesítés)

- **Argumentumokban használt metakaraktereket (közöttük a dzsókereket: * ? [ ]) a burok a lista/parancs végrehajtása előtt különlegesen kezeli.**
- **Ha a szavakban dzsókereket talál, azt a szót *mintának* (pattern) veszi.**
- **A minta behelyettesítődik alfabetikus sorrendű fájlnevek listájává, olyan nevekre, melyek a fájlnév-térben illeszkednek a mintára.**
- **Csak ezután hajtódik végre a parancs/lista.**

Az illeszkedés

- **„Szokásos” karakter önmagára illeszkedik …**
- **A ? egyetlen, bármely karakterre illeszkedik.**
- **A * tetszőleges számú, tetszőleges karakterre illeszkedik.**
- **A [...] illeszkedik egyetlen, valamelyik bezárt karakterre.**
- **A [!...] illeszkedik egyetlen, bármely, kivéve a ! utáni karakterre.**
- **stb., nézz utána!**

Példák

- **Tegyük fel, az aktuális jegyzékben van 4 fájl:**

a abc abc.d xyz

> ls * # ⇒ ls a abc abc.d xyz

> ls a* # ⇒ ls a abc abc.d

> ls [a]?? # ⇒ ls abc

> ls [!a]?? # ⇒ ls xyz

Vegyük észre, előbb megtörténik a behelyettesítés, csak azután hajtódik végre az ls parancs!

Vö: > rm * és DOS> DEL * Vigyázz: > echo *

A metakarakterek semlegesítése

- **Ezeket az sh kifejti, ezek miatt a parancsot átalakítja.**
- **Ha mégis szükségünk van ezekre a parancshoz, (pl. szűrőnek kellenek) semlegesítsük (quotázzuk) őket!**
- **Egyetlen karakter semlegesítése: \kar**
- **Több karakter semlegesítése:**
  - **‘karaktersor‘ # minden bezárt semlegesített.**
  - **“karaktersor“ # a paraméter és parancsbehelyettesítésen kívül (lásd később) minden semlegesített.**

Példák

```
$   a=valami
$   echo   $a
 valami
$   echo   ’$a’
 $a
$   echo ”*$a”
 *valami
$
```


A shell adatszerkezetei

- **Típusuk: szöveglánc (füzér). Csak numerikus karakterekből álló szöveglánc néha numerikus adatként viselkedhet: numerikus operációk hajthatók rajtuk végbe.**
- **Lehetnek:**
  - **Változók: ekkor van nevük, van pillanatnyi értékük. (Változó vagy paraméter?)**
  - **Konstansok: szövegkörnyezetből adódnak (lexikális szöveg-konstansok).**

A változók

- **A burok által definiált változók: nevüket nem mi választjuk meg. Lehetnek:**
  - **pozícionális változók (paraméterek),**
  - **egyéb a shell által definiált (speciális) változók.**
- **A felhasználók által definiált változók (nevüket kiválaszthatjuk):**
  - **a rendszergazda által definiáltak: konvencionális nevek!**
  - **Az egyes felhasználók által definiáltak: legyenek itt is konvencióink!**

A pozicionális paraméterek

- **Nevük: 0 1 2 3 4 5 6 7 8 9**
- **Pillanatnyi értékük: az aktuális argumentumok (a szavak)**
- **Emlékezz:**
  - **a 0. szó a parancs neve: a 0 nevű változó pill. értéke,**
  - **az 1. szó az 1. argumentum: az 1 változóban van.**
- **Annyi változó definiált, ahány szó van!**
- **Ha 9-nél több argumentum van: a shift paranccsal "eltolhatók", így kezelhetők!**

Néhány, a shell által definiált változó

Neve	Értéke	Kifejtése
#	A pozícionális paraméterek száma	$#
?	Az utolsó parancs visszatérési értéke	$?
$	A burok processz pid-je	$$
!	Az utolsó háttérben futó proc. pid-je	$!
-	A burok opciók	$-
*	A pozícionális paraméterek szólistája	$*
0	A parancs neve	$0

Néhány, a rendszer által szokásosan definiált, exportált változó

Neve	Értéke	Kifejtése
HOME	A bejelentkezési jegyzék	$HOME
MAIL	A levelezési fájlunk	$MAIL
PATH	Parancsok keresési ösvénylistája	$PATH
USER	A felhasználó neve	$USER
IFS	A burok opciók	$IFS
TERM	A terminál típusa	$TERM
PS1	Az elsődleges prompt	$PS1

Változódefiniálás

- **Szintaxis**
  - **$ valtozo=fuzerkifejezes [valtozo=fuzerkifejezes]**
- **Példa:**
  - **$ tmpfile=/tmp/valami**
  - **$ ures=**
- **Vigyázz! A következő nem jó! Miért?**
  - **$ valtozo = fuzerkonstans**

(Mert ez már 3 szó! Tilos a fehér karakter az = előtt és után!)

Hivatkozás a változókra, kifejtésük

- **A kifejtő operátor a $**

  $valtozo kifejtődik. Pl.:

  $tmpfile ⇒ /tmp/valami
- **Néha szükséges lehet az egyértelműsítéshez:**

  ${valtozo}

Pl:
```
$ val=valaki;  valaki=senki
$ echo  $valaki
senki
$ echo  ${val}aki
valakiaki
```


Példák

```
$ val=valaki;  echo $val
 valaki

$ val='echo valami'
$ $val
 valami

Írjuk ki az aktuális jegyzék tartalmát ls nélkül
$ val=*;      echo $val
 a  ab  abc  f1  f2 ...
```

A teljes változókifejtés

- **Itt: valt változónév, szo szövegkifejezés, parancs.**
- **Ha a : (colon) hiányzik, a 0 string-hosszúság nem ellenőrződik!**

${valt:-szo}	**Ha a valt definiált és nem 0 sztring, akkor kifejtődik. Különben a szo fejtődik ki.**
${valt:=szo}	**Ha a valt nem definiált vagy 0 sztring, akkor felveszi a szo-t. Ezután a valt kifejtődik.**
${valt:?szo}	**Ha a valt definiált és nem 0 sztring, akkor kifejtődik. Különben kiíródik a szo|default sztr.**
${valt:+szo}	**Ha a valt definiált és nem 0 sztring, akkor kifejtődik a szo. Különben semmi sem fejtődik ki.**

Parancsbehelyettesítés

- **A parancsok szokásosan az stdout-ra írnak.**
- **A ` ` (grave accent) közé zárt parancs fut és kimenete kifejtődik!**
- **Példa:**

```
$ valt=`wc  -l  < myfile.txt`        # a valt felveszi
                                     # a sorok számát

$ test `wc -l < myfile.txt` -gt 3 && cat myfile.txt
                  # ha több mint 3 sorból áll, írja ki
```


A változók érvényességéhez: a processzek környezete

- **Minden processznek - így a buroknak is - van környezete (environment)**
- **A környezet: *vált=szöveglánc* sorokból álló tábla**
- **Gyermek processz a környezetet örökli a szülőtől**
- **Mikor az sh processz indul, végigolvassa a *környezetét* és definiálja a benne található változókat**
- **További definíciók is lehetnek az sh életében, átdefiniálások is lehetnek. Ezek nem kerülnek a környezetbe (nem örökölhetők)**

Exportálás: többszintes öröklődés

- **Exportálással változót a környezetbe „teszünk” (ezzel lefelé öröklődővé tesszük)**

> export változónév-lista

Pl. a rendszergazda valahol “leírta”:

$ export MAIL HOME PATH ...

- **Exportálással csak leszármazottak örökölhetnek (fölfelé nem)**
- **Nem exportált változó nem látható a leszármazott processzekben, de visszatérve abba a burokba, amiben definiáltuk újra láthatóvá válik**