

4. gyakorlat. Unix alapok II.

Téma: Processzek: ps, kill, trap, bg, fg, wait. Szövegszerkesztés: vi, pico, joe. Midnigth Commander: mc., Sorok manipulációja).

Folytassuk az előző óra anyagát, lépünk be terminálkapcsolattal a kiszolgálóra

Folyamatok - processzek

A Unix és így a Linux rendszerek többfeladatos (*multitasking*) operációs rendszerek, időosztásos (*time sharing*) módszert használva több programot is képesek egyszerre futtatni. A kellően gyors váltások miatt a felhasználó számára mindez egyidejűséget kelt.

A párhuzamosan futtatott feladatokat (folyamatokat) kezelni is tudjuk: lekérdezni, leállítani, időzíteni, priorizálni.

A Unix minden futó feladathoz külön azonosítót **PID** (processz identifikator) rendel, hogy az egyszerre futó - akár azonos - programpéldányok között különbséget tudjon tenni. A PID egy egész szám, amely a gép bekapcsolásakor 1-ről indul és minden elindított feladat esetén eggyel növekszik. (A legnagyobb PID limit érték lekérdezhető az alábbi paranccsal: `cat /proc/sys/kernel/pid_max`)

Futó folyamatok lekérése

A futó folyamatokat a `ps` paranccsal kérhetjük le.

Az egyes mezők jelentése:

PID - a folyamat azonosítója

TTY - a vezérlő terminál azonosítója, jelen esetben ez a `ttyp0`

STAT - a folyamat állapota

TIME - a processz által eddig elhasznált processzor idő

A rendszerben futó összes folyamatot, a legbővebb információkkal Linux alatt a "`ps -aux`" opciókkal kérhetjük le. Ekkor a processzekről megtudjuk még tulajdonosukat, az időpontot amikor elindultak, valamint különféle erőforrás használati információkat (CPU, memória használat, a program mérete a memóriában).

`top`: Információk a processzekről és egyéb statisztikák a rendszerről. A `top` folyamatosan fut, és 5 másodpercenként frissíti a megjelenített információkat. Kilépni a "q" megnyomásával lehet.

Programok indítása

Programok indítása előtérben: A program az előtérben fut, amíg nem fejezi be futását addig a felhasználó az adott héjban (shell, burok) nem indíthat újabb programot. Az előtérben futó program birtokolja a billentyűzetet. A hagyományos elindítással, tehát alapértelmezés szerint a programok az előtérben futnak. Vagy az `fg <parancsnév>` használatával.

Programok indítása háttérben: Háttérben úgy tudunk futtatni programot, hogy `'&'` jelet kell gépelni a parancs után.

Feladat:

1. `sleep 20`

20 másodpercig vár, előtérben futtatva. A vezérlést 20 másodperc múlva kapjuk vissza.

2. `sleep 20&`
20 másodpercig vár a háttérben. Két számot ír ki, pl.: [1] 532 majd visszaadja a vezérlést. Az első szám az adott burok feladataazonosítója, a második a rendszeren érvényes egyedi feladatazonosító, PID. Majd 20 másodperc múlva, amikor a folyamat végzett, a burok üzenetet küld a felhasználónak pl.: [1] + Done. A felhasználó csak akkor kap értesítést, amikor a következő ENTER-t megnyomja.
3. `nohup <parancsnév>: (no hangup):` Ha egy parancs elé a `nohup` kulcsszót tesszük, az adott program futása nem szakad meg a felhasználó kijelentkezésekor. Kimenete hozzáfűződik a `nohup.out` állományhoz.
`nohup ls`
`more nohup.out`

Processzek kezelése

Az előtérben futó folyamatot be kell fagyasztani ahhoz, hogy később a háttérben, kívülről lehessen irányítani. `ctrl + z`.

Pl.: `sleep 100`, majd `ctrl + z`, továbbá kapunk egy folyamatazonosítót is, későbbi felhasználásra. Ezt követően a `ps-t` kiadva láthatjuk a még futó folyamatot.

- `fg`: Egy háttérben futó vagy befagyasztott folyamatot az előtérbe helyezhetünk vele.
`fg %1` vagy `fg sleep`
- `bg`: Egy befagyasztott programot a háttérben folytathatunk.
- `jobs`: belső feladatazonosítók lekérdezése. Kiírja továbbá a futó folyamatok állapotát is (Running, Stopped, stb.)
- `kill`: Folyamat bezárása. „Normál” felhasználó csak a saját folyamatait állíthatja le, a rendszergazda bárkiét. A `kill` parancs, paramétere lehet a folyamat száma vagy pedig a processz azonosítója (PID).

Hagyományos leállítás:

```
kill %1
kill 3954
```

Példa: A `yes` parancs `y` karakterek végtelen sorozatát küldi a kimenetére. Hogy ne kelljen az `y`-ok végtelen sorát látni irányítsuk át a kimenetet egy alkalmas helyre: `/dev/null` - egy ún. fekete lyuk: a beleirányított összes információt elnyeli: `yes > /dev/null`

Így kaptunk egy előtérben futó jobot. Hogy a háttérbe tegyük, a `"&"` jelet kell alkalmaznunk: `yes > /dev/null &`

Feladat: a `kill` paranccsal állítsuk le a futó folyamatot.

- Jelek: 0-15 közti számok, valójában egy eseményt jelöl egy szám
 - Egy jel küldése egy processznek: `kill -jel PID`
 - PL: 2 – `ctrl+c`, 9- `kill` ,feltétel nélküli befejezés
 - 15 jel az alapértelmezés, ez a szoftver befejezés

Feladat: Lépjünk be két terminál kapcsolattal, egyik ablakban indítsunk el egy szövegszerkesztőt, majd ezt löjük ki a másik ablakból! (`kill -15 PID`)

Szignálok: A Unix rendszer a folyamatok vezérlését a folyamatoknak küldött ún. szignálok (signals) segítségével végzi: a `^Z` billentyű például egy STOP szignált küld az előtérben futó processznek. Processzt kiölteni szintén szignál(ok) segítségével lehet: az előtérben futó program a `^C` (Ctrl-c) megnyomására egy INT szignált kap, amely rendszerint a program elhalálózását vonja maga után. Háttérben futó folyamatainkat a "kill" paranccsal irhatjuk ki: alapértelmezés szerint a "kill" egy TERM (terminate) szignált küld a megadott folyamatnak.

Ha más (nem TERM) szignált akarunk küldeni, a kill parancsot megfelelően paraméterezni kell, például a STOP szignálhoz: "kill -STOP pid". Ennek ugyanolyan hatása van, mintha az a folyamat az előtérben futna, és a ^Z-t nyomtuk volna meg: a folyamat felfüggesztett állapotba kerül. Folyamatot megölni még a HUP (hangup) és a KILL szignálokkal is lehet. (Az előbb látott nohup parancs ezen HUP szignál ellen teszi immunissá a folyamatot.) A sokféle látszólag azonos hatású szignál oka, hogy korántsem azonos hatásúak: például a HUP és a TERM szignálokat a folyamat felülbíráhatja, saját szignál-kezelő rutint állíthat be (így van ez az INT szignálnál is). Ezeket a szignálokat a folyamat kapja meg, és alapértelmezés szerinti kezelő rutinjuk lép ki. A KILL szignál hatására viszont a kernel öli meg a folyamatot, annak megkérdezése nélkül. Ezért nem probléma Unixban, ha egy folyamat "lefagy", végtelen ciklusba kerül: egy KILL szignál mindig megoldja a problémát.

Szignált csak saját processzeinknek küldhetünk (kivéve a root-ot, aki bármely processzel rendelkezhet). Az eddig felsoroltakon kívül még számos egyéb szignál van, megemlíthetjük még az ALARM szignált: a rendszert megkérhetjük, hogy megadott idő múlva küldjön egyet. Ezt használják időzítési célokra, többek között a "sleep" utasítás is így működik. De szignálokat használ a rendszer sok más egyéb, a folyamatot érintő rendszerinformáció közlésére is, de ezek főleg programozók számára érdekesek.

Processz erőforrás-felhasználás

- `time <parancsnév>`: Ha egy parancs elé a time kulcsszót írjuk, akkor a burok az indított program befejezése után statisztikai adatokat ad a feladat által használt erőforrásokról, kiírja, hogy mennyi idő alatt futott le és mekkora terhelést jelentett a rendszer számára.
pl.: `time ls`
- `strace <parancsnév>`: Hibakereséshez használatos parancs. Az indított program minden rendszerhívását kiírja.
pl.: `strace ls`
pl: `strace -p 16543` - a figyelt folyamat azonosítója

Csapdák (eseménykezelés):

`trap` parancs. Bizonyos jeleket lehet, címzés nélkül küldeni, `ctrl+c`, `ctrl+\` (3, quit)

Feladat: A `ctrl+c` leütésére írjuk ki: Hajrá ELTE!

Megoldás: `trap 'echo Hajrá ELTE!' 2`

Feladat: Szüntessük meg ezt a csapdát, „eseménykezelést”!

Megoldás: `trap '' 2`

Szövegszerkesztés

vi (vim)

Lényeges, mert mindenhol van! (szamalap-on a vim (improved) indul el!)

Parancsmód: `esc bill`.

1. mentés:
 - a. `:w`, `:x` – kilépés mentéssel
 - b. `:wq` – mentés, bezárás
 - c. `:q!` – kilépés mentés nélkül
2. karakter törlés: `x`
3. sortörlés: `dd`, 3 sor törlés: `3dd`
4. Beillesztés: `p`

5. Keresés: /mit

Szerkesztőmód: i, a, o

Segítség:

- <http://www.szabilinux.hu/vi/index.html>
- www.linuxvilag.hu/content/files/cikk/36/cikk_36_78_80.pdf

Feladat:

Készítsük el az alábbi fájlt vi.html néven a webes mappánkba (public_html):

```
<html>
<body>
  <p>Nem is olyan bonyolult ez a szerkeszto</p>
</body>
</html>
```

pico

Fontosabb parancsok a képernyőn

joe

Help: ctrl+k, h

Feladat: Módosítsuk például a címkét az index.html állományban!

Midnight Commander (mc)

Fontos, többcélú fájlmenedzsment program, akár szövegszerkesztőként, FTP kliensként is használható. Hasonlít a sokak által ismert Norton Commanderhez. Kétpaneles, de mindkét panel a távoli gépre vonatkozik!

Sorok manipulációja

A szűrők később visszatérnek a scriptes részben! Elegendő arra utalni, hogy milyen fontos feladatot látnak el. Nem kell ezeket a feladatokat megoldani, elég egy-két példa!

Parancsként és szűrőként is működnek pl. wc, grep, cut, sort

Feladat:

1. Számoljuk meg az aktuális könyvtárban lévő bejegyzések számát. Készítsük el a megoldást parancsként és szűrőként is a megoldást!
2. Listázzuk ki az aktuális könyvtár alkönyvtárait! (A könyvtáraknál d az első karakter – grep ^d, cut)
3. Listázzuk ki a bejelentkezett felhasználók azonosítóját és csak azt! (who, cut)
4. Listázzuk ki egy tetszőleges szöveges fájl sorait abc sorba rendezve!
5. Cseréljük ki egy fájlban lévő karaktereket a nagybetűs párjukra!
6. Számoljuk meg, hogy egy fájlban hány olyan sor van, amelyik az alma szót tartalmazza és semmi mást! (tr, grep ^alma\$, wc)
7. Csak az egészeket tartalmazó sorokat engedje át a szűrő (grep ^[+-]?[1-9][0-9]*\$)