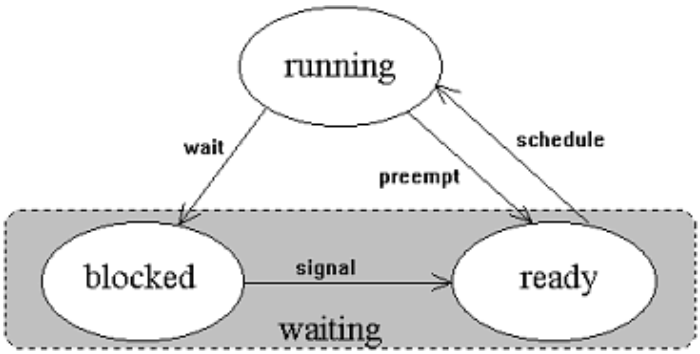
Folyamatkezelés

A Linux többfeladatos (multitask) és többfelhasználós (multiuser) rendszer. Ebből következik, hogy akár egy felhasználó egy időben több programot is futtathat. Az elindított program a processz, azaz folyamat, más megfogalmazásban egy végrehajtható fájl "élő" változata. Gyakran taszknak is nevezik. A folyamatok jól definiált hierarchiát alkotnak. Minden folyamatnak pontosan egy szülője (parent) van, és egy vagy több gyermek folyamata (child process) lehet. A folyamat hierarchia tetején az init folyamat helyezkedik el. Az init folyamat az első létrehozott felhasználói folyamat, a rendszer indulásakor jön létre. Minden felhasználói folyamat az init folyamat leszármazottja. Néhány rendszer folyamat, mint például a swapper és a page daemon (a háttértár kezelésével kapcsolatos folyamatok), szintén a rendszer indulásakor jön létre és nem az init folyamat leszármazottja. Ha egy folyamat befejeződésekor még léteznek aktív gyermek folyamatai, akkor azok árvákká (orphan) válnak és azokat az init folyamat örökli - majd egyben meg is szünteti azokat. A Linux minden egyes feladathoz két számot (PID, process identificator - feladat azonosítót és a PPID, parent process identification - szülő azonosítója) rendel. A rendszer a PID-et automatikusan növeli. Az init folyamat PID-je 1.

Az "életre keltett" folyamatok szekvenciálisan hajtódnak végre, azaz a felhasználó csak akkor kapja vissza a készenléti jelet, ha a végrehajtás befejeződött - alapvetően szinkron módon működik. A processz futhat előtérben (billentyűzetet és a képernyőt magához ragadva), és háttérben (manuálisan is elő lehet idézni, melynek formája:`parancsnév&`). Ha egy előtérben futó folyamatot szeretnénk háttérbe helyezni, a suspend funkcióhoz rendelt karakterrel (ez rendszerint a Ctrl+Z) tudjuk az előtérben futó folyamatot felfüggeszteni. Ezután pedig a megfelelő parancsokkal tudjuk folytatni a futását, háttérbe helyezni, végleg leállítani, stb.).

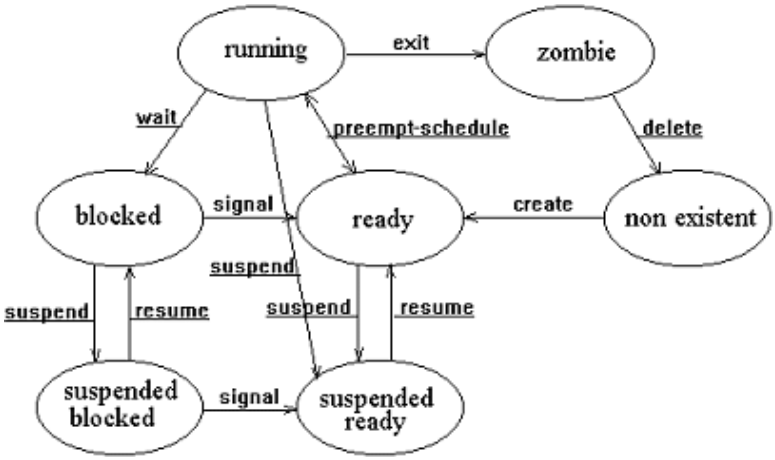
Minden processz önálló entitás a saját programszámlálójával és kontextusával. Lehet közöttük együttműködési kapcsolat, akár szinkron, amikor az egyik processz készít valamilyen output-ot, ami a majd másik processz bemenete lesz, illetve aszinkron, amikor már két futó folyamat kommunikál. Szinkron esetben a két processz futásának relatív sebességétől függően előfordulhat, hogy a második processznek várnia kell, amíg az első a kimenetét elkészíti. A második u.m. blokkolt amíg az inputja elkészül. Kérdés merülhet fel, hogyan "billen" ki ebből az állapotból a blokkolt processz. Másrészt az is előfordulhat, hogy egy processz ugyan nem vár semmire, tehát futhatna, de az operációs rendszer egy másik processznek adja át a CPU-t: ekkor is "vár" a processzünk, most a CPU-ra, ezt az állapotát feltétlenül meg kell különböztetni az inputra való várakozástól. Azt mondhatjuk, hogy a processzek - életük során - különböző állapotokban (state) lehetnek, az állapotok között különböző állapotátmenetek lehetségesek. A legegyszerűbb és legáltalánosabb állapot és állapotátmenet diagram a következő:



Egy speciális háttérfolyamat a daemon. Ezek nagy részét a Linux rendszer már a rendszerbetöltéskor elindítja. Számos démon fut a háttérben és figyeli, pl. a lokális hálózatba belépőket, a nyomtatási kérelmeket, stb. Pl.: inetd (tcpd), ftpd, httpd. Tipikusan a szolgáltatásokkal lehet őket azonosítani. Talán ezért is lehet őket a /sbin mappa alatt található `service` utasítással vezérelni. Inaktív állapotban várakoznak arra, hogy a szolgáltatásuk igénybevételre kerüljön, ekkor aktív állapotba kerülnek, kiszolgálják a kérést, majd visszamennek inaktív állapotba.

Egy másik speciális helyzetű folyamat a zombie, a már halott (leállt), de még a rendszerből el nem tűnt folyamat: akkor lehetséges ez, ha a gyermek folyamat már kilépett, de a szülő még nem fogadta a gyermek visszatérési értékét - még nem vett tudomást gyermeke haláláról (befejeződéséről).

A rendszeren belüli egyes állapotok és a közöttük fellelhető állapotátmeneteket az alábbi ábra szemlélteti:



Folyamatkezelő parancsok

ps

A Process Sate parancs processzusok állapotát jeleníti meg.

Szintaktika:

```
ps [kapcsolók]
```

Kapcsolók:

- -e : az összes folyamat megjelenítése
- -f: részletes lista
- -u username : megjeleníti az adott felhasználó összes folyamatát

Mezők jelentése:

- PID : a folyamat azonosítója
- TTY : a vezérlő terminál azonosítója
- STAT : a folyamat állapota
- TIME : a processz által eddig elhasznált processzor idő

- CMD : a processz neve

Példa a ps parancs használatára:

```
adamkoa@it:~$ ps
PID   TTY   TIME     CMD
15573 pts/4 00:00:00 bash
16407 pts/4 00:00:00 ps
adamkoa@it:~$
```

pstree

Az initből induló folyamathierarchiát lehet a parancs segítségével megtekintetni fa szerkezetű ábrázolásban.

nohup

Mikor a rendszerből kijelentkezünk (azaz a bash bezáródik) minden gyerekfolyamatát a rendszer automatikusan kilövi. Lehetőségünk van azonban arra is, hogy egy folyamatot immúnissá tegyünk kilépésünkre. Hosszan, több óráig, több napig futó programokat a nohup paranccsal indíthatunk.

Példa a nohup parancs használatára:

```
adamkoa@it:~$ nohup program
adamkoa@it:~$
```

top

A top a kill és a ps parancs egyesített változata mely folyamatosan futva listázza az éppen aktív folyamatokat, információt nyújt a rendszer állapotáról és terheltségi mutatóiról illetve lehetőséget ad szignálok küldésére folyamatok számára. A top-ot a parancssorban kiadott top utasítással indíthatjuk.

A Linux rendszer a folyamatok vezérlését a folyamatoknak küldött ún. szignálok segítségével végzi: a Ctrl+Z billentyű például egy STOP szignált küld az előtérben futó processznek. Igen sok (kb. 60 db) szignál létezik, ezek közül csak néhányat tárgyalunk. Processzt megszüntetni szintén szignál(ok) segítségével lehet: az előtérben futó program a Ctrl+C megnyomására egy INT szignált kap, amely rendszerint a program elhalálozását vonja maga után. Háttérben futó folyamatainkat a kill paranccsal állíthatjuk le.

kill

A kill a nevével ellentétben nem csak folyamatok megölésére használható, hanem tetszőlegese signalt küldhetünk vele bármely folyamatnak melynek tudjuk a tudjuk a PID számát és rendelkezünk a folyamat kezeléséhez megfelelő jogokkal. Alapértelmezés szerint (signal paraméter nélkül használva) a kill egy TERM (terminate) szignált küld a megadott folyamatnak.

Szintaktika:

```
kill [signal] [PID]
```

Példa a kill parancs használatára:

```
adamkoa@it:~$ ps
PID TTY STAT TIME COMMAND
```

```
310 pp0 S    0:00 -bash
313 pp0 R    0:00 ps
321 pp0 R    0:00 find -name= doksi

adamkoa@it:~$ kill 321

adamkoa@it:~$ ps
PID TTY STAT TIME COMMAND
310 pp0 S    0:00 -bash
334 pp0 R    0:00 ps
adamkoa@it:~$
```

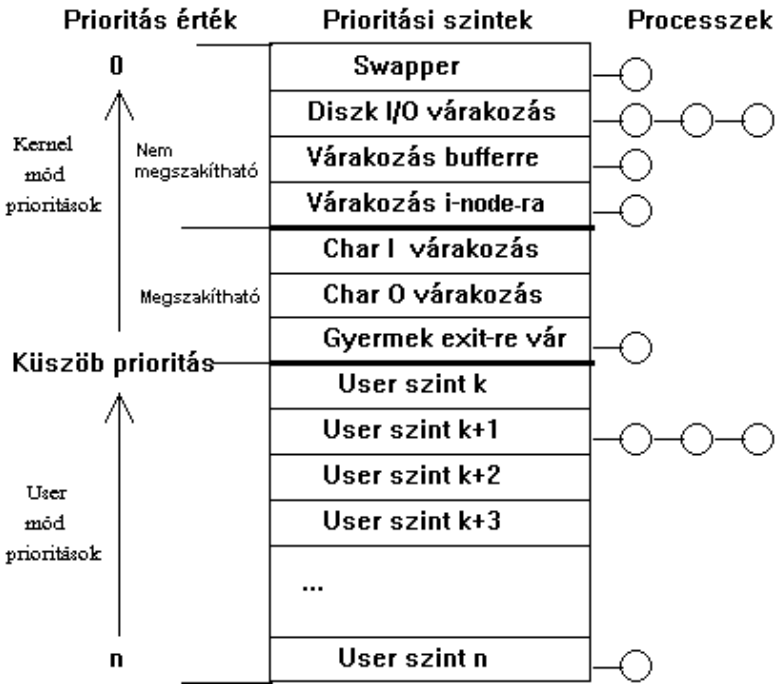
Ha más (nem TERM) szignált akarunk küldeni, a kill parancsot megfelelően paraméterezni kell. Folyamatot megölni még a HUP (hangup) és a KILL (9-es) szignálokkal is lehet. (a nohup parancs ezen HUP szignál ellen teszi immúnissá a folyamatot.) A sokféle látszólag azonos hatású szignál oka, hogy korántsem azonos hatásúak: például a HUP és a TERM szignálokat a folyamat felülbírálhatja, saját szignál-kezelő rutint állíthat be. 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 processzről rendelkezhet). Fontos még az ALARM szignál. Ezzel a rendszert megkérhetjük, hogy megadott idő elteltével küldjön egy jelet. Ezt használják időzítési célokra, többek között a sleep utasítás is így működik.

Prioritás

A prioritás (az ütemezési - scheduling - prioritás) azt szabja meg, hogy ha több folyamat is van egyszerre futóképes állapotban, akkor a kernel milyen arányban ossza meg a rendelkezésre álló CPU időt az egyes processzek között. Unixban a prioritás számszerű értéke minél kisebb, annál több CPU időt fog kapni a folyamat. Prioritás értéke 19-től -20-ig terjed és a negatívabb érték magasabb prioritást jelent. A processzek prioritását a "top" paranccsal vagy a "ps" parancs -l opciójával a PRI oszlopban lehet megnézni.

Minden folyamat három prioritással rendelkezik: egy alapprioritással (base priority), amely állandó, egy ütemezési prioritással (sceduling priority), amely a program futásakor nő, és egy ún. "nice" prioritással, amely (bizonyos határok között) felhasználó által változtatható. Ütemezéskor e három érték bizonyos szabályok szerint képzett összegét használja a rendszer: az ütemező algoritmus döntési pontján mindig a legalacsonyabb összeggel rendelkező processz kapja meg a vezérlést (ezért kell ebbe az összegbe az elhasznált CPU idővel növekvő tagot is tenni: egyébként mindig csak a legmagasabb prioritású folyamat futna).

Az alábbi ábra mutatja, hogy miként is tudjuk elképzelni ezeket a különböző prioritásokat:



A "`nice -n növekmény parancs`" utasítás szolgál arra, hogy a "parancs"-ot a megnövelt nice prioritás értékkel futtassuk. Erre akkor lehet szükség, ha valami számításigényes, hosszan futó programot indítunk, de nem akarjuk jelentősen lassítani az interaktívan dolgozók munkáját. Ezt a "nice" értéket egyébként a "top" "r" parancsával is megváltoztathatjuk. Nem privilegizált felhasználó csak növelni -azaz gyengíteni- tudja folyamatai nice értékét (illetve a visszacsökkentéskor nem tudja az induló érték alá csökkenteni), a root felhasználó természetesen tetszőlegesen állíthat prioritást.

Már futó processz esetén a renice parancs segítségével tudunk a prioritáson változtatni.

Előtér, háttér

Háttérfolyamatot előtérbe hozni az **fg** paranccsal tudunk. Ha egy másik programmal szeretnénk foglalkozni, de azt akarjuk hogy az előtérben lévő folyamat tovább fusson a háttérben, akkor a Ctrl+Z billentyűkombinációval megállíthatjuk (ekkor várakozó, "stopped" állapotba kerül), majd háttérbe helyezni a **bg** paranccsal tudjuk. Ha a folyamat futásképes, (nem vár mondjuk terminál inputra) akkor a háttérben tovább fog futni. Kilépéskor, ha vannak még háttérben futó vagy várakozó folyamataink, a rendszer erre figyelmeztet a `"You have running jobs"` vagy `"You have stopped jobs"` üzenettel. Ha közvetlenül ez után még egyszer kiadjuk a logout parancsot, a shell leállítja a háttérfolyamatokat és kiléptet bennünket a rendszerből. Ha az fg és bg parancsokat argumentum nélkül használjuk, mindig a legutoljára hivatkozott folyamatra vonatkoznak. Ettől eltérő esetben hivatkozhatunk a feladatra a job azonosítójával, amit a `jobs` parancs ad meg, vagy hivatkozhatunk a nevével is. Minkét esetben egy % jellel kell bevezetni a paramétert. (Grafikus felület esetén erre a legjobb szemléltető példa az xeyes nevű program többszöri megnyitása és ezek szabályozása.)

Ütemezett végrehajtás

A UNIX multitaszking képessége nem korlátozódik csak a jelenben a futó folyamatokra. Tartalmaz programokat, amelyek lehetővé teszik a programok ütemezett futtatását, akár egyszeri, akár ismétlődő időközönként. Egy adott időben az `at` program segítségével indíthatunk el folyamatokat, ismétlődő esetekben pedig a `corntab` alkalmazással.

at

Az `at` parancs lehetővé teszi a programok,parancsok vagy shell skriptek egy jövőbeli dátum és időben történő futtatását. Például ha e-mailben szeretnénk egy fájl tartalmát elküldeni, vagy a find segítségével szeretnénk egy keresést indítani, akkor amikor a rendszer terheltsége alacsony, például egy hajnali órában.

Az at használatához az alábbi lépéseket hajtsuk végre:

- Adjuk meg az at parancsot egy időspecifikációval,ahol az idő meghatározása lehet:

  $at 10:30am today

  $at midnight

  $at 12:01 1 jan 2012

  Lásd még a man oldalt további példákért.

- Az at promptjánál (`at>`) adjuk meg a végrehajtandó parancsot.
- Több utasítás megadásához üssünk Enter-t, vagy CTRL-D-t a befejezéshez.

Ezután egy azonosító rendelődik az ütemezett feladathoz és bekerül a végrehajtási sorba. A sor állapotát az `atq` paranccsal tekinthetjük meg, ha pedig el szeretnénk távolítani egy ütemezett feladatot, akkor az `atrm [job#]` paranccsal tehetjük meg.

Ha csak egy parancsot szeretnénk futtatni, akkor azt megtehetjük az interaktív mód használata nélkül is: `$at [időspec][szkriptfájl neve]`. Például a `$at midnight whoison` segítségével megnézhetjük kik dolgoznak még éjfélkor is a gépen.

crontab

A crontabbal lehetőségünk nyílik időzített programfuttatásra, melynek kimenetéről e-mailben kapunk értesítést.

Mindig csak egy-egy felhasználóra vonatkozó crontabot lehet módosítani. Csak a superuser adhat meg a magáétól különböző felhasználónevet, illetve más crontab könyvtárat a parancshoz. Általában a -e opció jeleneti a saját crontab-unk módosítását. A crontab-ok módosításához a vi vagy a joe szövegszerkesztőt, használja a parancs.

Az egyes mezők tartalmazhatnak időpontot, időintervallumot, skip faktoros időintervallumot, szimbolikus intervallumot a hét napjaira, illetve az év hónapjaira, valamint további részintervallumokat vesszővel elválasztva. A crontab file -ban lévő üres, vagy kettős kereszttel kezdődő sorokat a parancs nem veszi figyelembe. Ha megadtuk a hét és a hónap egyik napját is, akkor a crontab bejegyzés le fog futni minden héten a megadott napon, valamint minden hónapban a megadott napon (a két feltétel vagy kapcsolatát képezzük.).

Példa a crontab parancs használatára:

```
# m h dom mon dow command
# PERC ÓRA NAP HÓNAP AHÉTEGYNAPJA PARANCS # MIN HOUR DAY MONTH DAYOFWEEK COMMAND
# minden nap reggel 6:10-kor
10 6 * * * date

# minden második órában az óra végén
0 */2 * * * date

# minden második órában reggel 11-től este 7-ig , valamint este 8-kor
0 23-7/2,8 * * * date

# este 11-kor negyedikén, valamint minden hétfőn, kedden, és szerdán
0 11 4 * mon-wed date 0 11 4 * mon-wed date

# január elsején délután 4-kor
0 4 1 jan * date 0 4 1 jan * date

# óránként egyszer, és minden kimenet a log file -ba menjen
0 4 1 jan * date >>/var/log/messages 2>&1 0 4 1 jan * date >>/var/log/messages
2>&1
```

Referencia kézikönyv

Folyamatok

```
command &
        # command futtatása a háttérben
        fg "sorszám"    : felélesztés, majd előtérben futás
        bg "sorszám"    : felélesztés, majd háttérben futás

Ctrl+Z
        # Előtérben futó process (pl mcedit) háttérbe helyezése.

Ctrl+C
        # Félresikerült/megakadt process bezárására használható billentyű
kombináció

jobs
        # Háttérben futó programok kiíratása
        # Az itt megkapott értékek használhatóak az fg, bg parancsoknál.


command1 && command2
        # command1 sikeres futását követően command2 is lefut.
        # Amennyiben command1 visszatérési értéke nem 0 (tehát sikertelen), úgy
az utána
        # soron következő parancs(ok) nem fut(nak) le


                letix@microserver:~/test$ ls && echo masodik parancs sikeres
                dir1  dir2
                masodik parancs sikeres
                letix@microserver:~/test$ ls dir3/ && echo masodik sikeres
```

ls: dir3/ nem érhető el: Nincs ilyen fájl vagy könyvtár

```
top
        # Futó folyamatok kiíratása

        h       : help
        u       : felhasználókra szűrhető lista
        z,b     : a táblázat színezése, illetve kivastagítása
        Z,B     : a táblázat színeinek, kivastagíthatóságának módosítása
        l,t,m   : a lista fejléce, terhelési adatok, egyebek jeleníthetőek
meg/kapcsolhatóak ki

pidof folyamat
        # A folyamat azonosítóját adja meg (PID - process ID)

ps
        # Futó folyamatok kiírása
        -u pisti : pisti felhasználó által futtatott folyamatok
        aux      : minden folyamatot kiír, szinte minden információval
        alxww    : minden folyamatot, még több infóval (pl.: PPID)
        -t1      : tty1-es terminál kilistázása
        f        : erdő szerű megjelenítés
        l        : kiírja a folyamatok PID-jét és PPID-jét is. (parent's process
identifier)
        ww       : a programok parancssori kapcsolóit írja ki.
        fax      : fa-szerű struktúrában jeleníti meg a folyamatlistát

        -o user,pid,ppid,start_time,uid,%cpu,%mem,cmd
        # felhasználónév, processid, parent processid, kezdési idő, CPU, MEM,
kapcsolók
        # formában adja meg a folyamatokat.
```

USER PID PPID START UID %CPU %MEM CMD
letix 19284 19283 10:39 1000 0.0 0.1 -bash
letix 22095 19284 14:33 1000 0.0 0.0 ps -o
user,pid,ppid,start_time,uid,%cpu,%mem,cmd

```
pstree
        # Folyamat struktúra fa-szerű ábrázolása.

pgrep
        # Folyamatlista szűrése adott minta alapján
        -u user           : megadja user nevében futtatott folyamatok PID-jét
        -u user screen    : megadja user nevében futtatott screen folyamat PID-
jét
        -lu 0             : megadja a 0 UID-el rendelkező (root) felhasználó
folyamatainak nevét és PID-jét

pkill
        -9 -u user screen : user nevében futtatott screen folyamat erőltetett
leállítása
                            hasznos lehet, ha többen futtatnak screen-t



kill "pid"
        # Folyamat leállítása PID szerint
        -1          : jelentése SIGHUP. A folyamat bezárása, config fájl
beolvasása, folyamat újraindítása
        -9          : jelentése SIGKILL, folyamat erőltetett bezárása. (Csak
végszükség esetén)
        -15         : jelentése SIGTERM, szabályos programleállítás
```

-l : a teljes signal táblázat kiíratása. Ezek a jelek küldhetőek folyamatoknak
-s "signal" : -l átal megadott táblázatban található signal-ok küldhetőek adott processnek.

letix@microserver:~$ yes > /dev/null &
[1] 22268
letix@microserver:~$ kill -s 6 22268
[1]+ Félbeszakítva yes > /dev/null

killall command
az összes fehasználó által futtatott "command" nevű folyamat leállítása

nice
Priorítása lekérdezése, beállítása
-n --20 program : a rendszer a legtöbbet ezzel a programmal fog foglalkozni. (+19-től -20 ig)

renice -10 1124
1124-es folyamat -10-es prioritásra állítása

init
Futtatási szint beállítása (run levels)
Az egész rendszer állapotjelzője, ami meghatározza, mely szolgáltatások működnek, vagy épp indulnak el
rendszerindításkor. A futási szintek számokkal kerülnek azonosításra.
Definiálhatóak különböző szintek annak függvényében, hogy milyen jellegű munkára lesz használva a gép.
Például ha X-el, bluetooth-al és egyéb erőforrás igényes alkalmazásokkal lesz használva, úgy létrehozható
egy 5-6-os init szint. Ezt a szintet az inittab-ban szükséges beállítani alapértelmezettként induláshoz,
de akár a rendszer futása közben is módosítható az aktuális szint. -> Akár több szolgáltatás is indítható vagy
leállítható egy paranccsal.
#
Init szintek
---

0 : kikapcsolás
1 : single-user mód (speciális rendszeradminisztrációs funkciókra)
2-5 : multi-user mód, (normál működés)
6 : reboot

Új szolgáltatás hozzáadása

Program bemásolása /etc/init.d-be, majd erről egy link létrehozása a kiválaszott
init szint könyvtárába (pl.: rc2.d) Csak akkor indulnak el, ha S betűvel kezdődnek.
A kezdőbetű után levő számok az induló folyamatok sorrendjét befolyásolják.
#
/etc/inittab -ban állítható be a gép alapértelmezett indulási init szintje.

The default runlevel.
*id:**2**:initdefault:*

```
        # Daemonok kezelése
        # --------------------

        /etc/init.d/daemon_nev start    :   daemon indítása
        /etc/init.d/daemon_nev stop     :   daemon leállítása
        /etc/init.d/daemon_nev restart  :   daemon újraindítása
        /etc/init.d/daemon_nev status   :   daemon status infók kiíratása


runlevel
        # Megadja, hanyas init szinten voltunk és vagyunk. (kimenet pl.: N 3 ,
vagy 3 2.) típusú.
        # 3 N jelenti, hogy 3-ason voltunk és vagyunk, 3 2 pedíg hogy 3-ason
voltunk 2-esen vagyunk

fuser
        # Folyamatok azonosítása nyitott file-ok vagy process-ek alapján
(érdemes root-ként futtatni)
        -v .            : aktuális felhasználó folyamatai
        -v -n tcp 80    : mely folyamat használja a TCP/80-at?
        -vm  /mnt/test  : megadja azon folyamatot, mely fogja /mnt/test mappát.
        -vmk /mnt/test  : kilövi azon folyamatot, mely fogja /mnt/test-et
        -v /var/run/mysqld/mysqld.sock  :  mely folyamat használja a mysqld.sock
socketet?


                root@microserver:/home/letix# fuser -v -n tcp 10000
                                     FELHASZNÁLÓ  PID  HOZZÁFÉRÉS PARANCS
                10000/tcp:                root  1764 F....      miniserv.pl
                root@microserver:/home/letix# mlocate miniserv.pl
                /usr/share/webmin/miniserv.pl




shutdown
        # Kikapcsolás
        -h now     : Azonnali kikapcsolás (időt is megadhatunk)
        -h 12:00 & : A gép kikapcsolása 12:00-kor.
        -c         : Az időzített kikapcsolási folyamat megszakítása
        -r 0       : Azonnali újraindítás

        -h `date --date "now + 30 seconds" "+%H:%M"`: aktuális dátumhoz képest
30 másodperccel későbbi leállítás kezdeményezése
```

Időzített parancsfeldolgozás

```
at
        # Megadott időpontban futtathatunk programokat
        -f todo 23.59 : előre megírt parancsainkat (todo fájlban) 23:59-kor
lefuttatja az at.
        # Idő formátumok
                # 13.13 01.01.02 : 13 óra 13 perc, 2002, január 1
                # 2pm tomorrow : honlap délután 2
                # 1am Sun : hajnali 1 óra, vasárnap

atq
        # Az at várakozási sorrendjét ismerteti

atrm pid
```

```
        # Az at várakozási sorából való eltávolítás, Process ID alapján

cron
        # A Linux feladatütemezője
        # /etc/cron.d ; /etc/cron.daily ; /etc/cron.weekly
        crontab -l : kilistázza a belépett user beállított ütemezéseit
        crontab -e : Editáljuk a belépett user ütemezéseit


        Cron job paraméterezése
        -----------------------

        Perc        Óra    Hónap napja      Hónap            Hét napja
Parancs
        (0-59)     (0-23)    (1-31)    (1-12 v. Jan-Dec) (0-6 v Vas.-Szo.)
Parancs


        Példák
        ------

        0            2         12             *              0,6
ping 192.168.1.1

        Minden hónap minden szombatján és vasárnapján amelyek 12.-ére esnek,
2:00-kor megpingeli a címet.


        30          10          *             *               *
ping 192.168.1.1

        Minden nap, 10:30 kor megpingeli a címet.


        00       1-8,12-17      *             *               *
ping 192.168.1.1

        Minden nap, 1:00-től 8:00-ig és 12:00-től 17:00-ig minden óra 0.
percében fut


        Alkalmazhatunk például vesszőt is, az első rublikába írva "0,30" jelenti
minden óra
        0. illetve 30. percét, illetve intervallum is megadható.


        További példák
        --------------

        @reboot parancs : A következő indításnál lefutó parancs.
        @weekly         :"0 0 * * 0" : Hetente egyszer fut le, vasárnap
éjfélkor.
        @daily          :"0 0 * * *" : Naponta egyszer fut le, éjfélkor.
        @midnight       :"0 0 * * *" : éjfélkor, ekv. az előzővel.
        @hourly         :"0 * * * *" : Minden egész órakor fut le.



        Cron job hibacsatorna /dev/null-ba irányítása
        ---------------------------------------------

        0 1 5 10 * /path/script.sh  >/dev/null 2>&1
```

```
        # amennyiben szükséges, hogy az adott job hibacsatornája ne árassza el a
/var/log-ot, úgy a fenti
        # kivastagított eljárást szükséges alkalmazni


date
        # Dátum kiíratása vagy beállítására használatos program
        +%F             : 2011-11-04 formátumban írja ki az aktuális dátumot.
        +%Y%m%d         : 20111104 formátumban írja ki az aktuális dátumot.
        -d-2day +%F     : 2011-11-02 formátumban írja ki a 2 nappal korábbi
dátumot.
        110411532011    : Beállítja a dátumot November 04., 11:53-ra, 2011-ben.
(Honap Nap Ora Perc Ev)

        -d '+3 hour' +%Y.%m.%d" "%H:%M.%S : az aktuális dátumhoz kéepst 3 órával
későbbi dátumot adja meg 2017.01.05 14:07.12 formátumban.


sleep 5 parancs
        # 5 Másodperc múlva indítja a "parancs"-ot.

schedutils
        # Linux rendszer ütemező

screen
        # Ablakkezelő, virtuális terminál emulátorral.
        # Háttérbeli programok futtatásához alkalmazható program.

        screen                  : Egy új VT-t (virtuális terminált) indítható
        Ctrl+a+d                : bill. kombinációval tehetjő háttérbe.
        screen -ls              : screen-ek listázása
        screen -d -m -S name    : elindítja a háttérben a screen-t "name"
névvel
        screen -x name          : name nevű terminál hozható előtérbe
        screen -R PID           : paranccsal lehet előtérbe hozni. (PID -
processID, ps aux-al megnézhető)


        # SCREEN-ben kiadható billentyű kombinációk
        -----------------------------------------

        Ctrl+a+d                : bill. kombinációval tehető háttérbe.
        Ctrl+a+?                : előhozható a legfontosabb bill.
kombinációkat
        Ctrl+a+c                : új ablak ltérehozása
        Ctrl+a+p VAGY n         : előző vagy következő ablakra ugrás.
(previous, next)
        Ctrl+a :kill            : aktuális screen lelövése
        Ctrl-a :acladd USER     : USER nevű felhasználó engedélyezése
becsatlakozásra (lásd MULTIUSER MODE)


        # MULTIUSER MODE
        # --------------
        #
        # A screen többfelhasználós üzemmódja
        # Amennyiben kontrollált körülmények között szükséges beengedni valakit
a gépre közös shellt
        # használva, úgy az alábbi lépéseket kell alkalmazni

        screen
                # screen indítása a kiszemelt többfelhasználós gépen

        Ctrl+a :multiuser on
```

```
                # Ezt begépelve aktiválható az adott session-ön a
többfelhasználós mód.
                # Másik oldalnak szükséges ismernie az adott user
nevét/jelszavát, illetve SSH-n be kell tudnia
                # csatlakozni a gépre. Ha mindez megvan, és SSH-n bejött.:

        screen -x
                #Voila!


time command
        # A command lefutási idejét méri

command &
        # command háttérbeli futtatása


command1 && command2
        # command1 visszatérési értékének függvényében command2 is lefuthat. (ha
command1 sikeres volt)
```