Brukerhåndtering av prosesser i Linux

Oppstart av prosesser fra shellet

- Prosesser startes ved å gi en kommando som svarer til navnet på det eksekverbare programmet
- Hvis programmet ikke ligger i stående katalog:
  - Katalogen der programmet ligger må enten være lagret i `PATH`-variabelen, eller søkesti må angis eksplisitt
- Hvis programmet ligger i stående katalog:
  - Angi `./` foran kommandonavnet, hvis ikke "`.`" ligger i listen av kataloger i `PATH`
- Husk å redirigere input/output hvis det trengs:

```
./somescript < my-data > my-results
```

PID: Identifisering av prosesser i Linux

- Alle prosesser har en unik PID – prosessidentifikator:
  - Et heltall – prosessene nummereres fortløpende etterhvert som de opprettes
  - PID-nummere gjenbrukes når de blir ledige
  - Den første prosessen som startes, `init`, har PID lik 1
- Prosesser har også:
  - Parent ID (PPID), som er PID til forelderprosessen
  - User ID (UID), en bruker som eier prosessen
  - Group ID (GID), tilhørighet til en brukergruppe

Prosesseier og -gruppe

- Alle Linux-brukere har en UID og en GID
  - UID: Et heltall som er unikt for brukeren
  - GID: Et heltall som angir brukergruppen man tilhører
- Prosessen *arver* UID og GID fra bruker som starter den
- Det finnes tilfeller (f.eks. endring av passord) der man av sikkerhetshensyn ønsker at bruker *ikke* skal være eier av prosessen, og/eller at gruppetilhørigheten skal endres
- Kan sette spesielle tilgangsrettigheter (execution bit) på den eksekverbare filen, for å styre eierskapet og gruppe for prosessen når et program kjøres *

*: Se avsnitt 4.3.1 i læreboken for detaljer

Forgrunns- og bakgrunnsprosesser

- Linux-programmer som startes på vanlig måte fra kommandolinjen, kjøres “i forgrunnen”:

```
$ program_navn
```

- Shellet oppretter (spawner) en ny prosess med `fork`, og bruker deretter systemkallet `wait` – shellet blir suspendert og må *vente* til den nye prosessen er ferdig

- Ved å bruke tegnet `&` (“og-tegn”) etter en Linux-kommando, kjøres kommandoen “i bakgrunnen”:

```
$ program_navn &
```

- Shellet lager nå i stedet ny prosess ved bare å bruke systemkallet `fork` – suspenderes *ikke* med `wait`

Kjøring av bakgrunnsprosesser

- Prosessen kjøres *samtidig* med shellet
- Shellet venter ikke til kommandoen er ferdig, men er klar for å motta en ny kommando med en gang
- Shellet vil skrive ut et *jobbnummer* og PID for bakgrunnsprosessen som opprettes
- Bakgrunnsprosesser kjøres med lavere prioritet
- Kan ikke kommunisere interaktivt med shellet eller med bruker

Håndtering av bakgrunnsprosesser

- Kan bare være én aktiv forgrunnsprosess
- Prosessen i forgrunnen kan *suspenderes* (settes i wait-tilstand) ved å taste `CTRL-Z`
- Kommandoen `bg` vil starte opp igjen *siste* suspenderte jobb som en bakgrunnsprosess
- For å se liste over alle prosesser i bakgrunnen, med status og jobbnummer: `jobs`
- Kommandoen `fg` *`jobbnummer`* brukes for å flytte prosesser fra bakgrunn til forgrunn

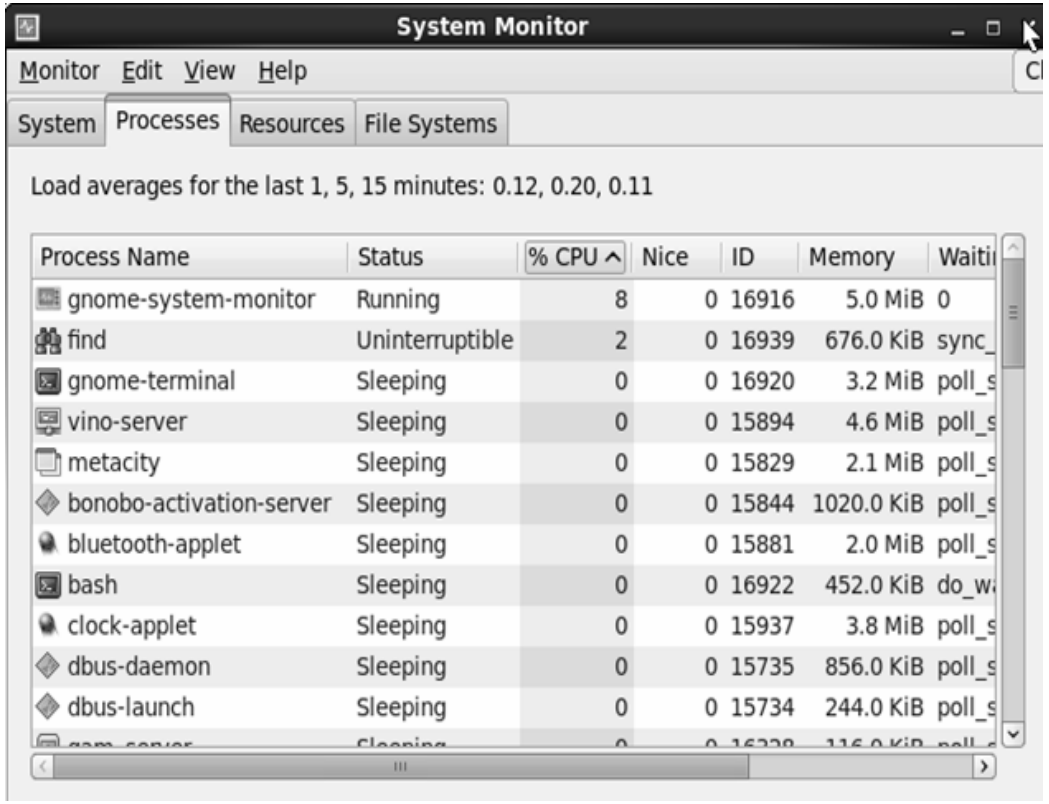
Prosessinformasjon og -status i Linux

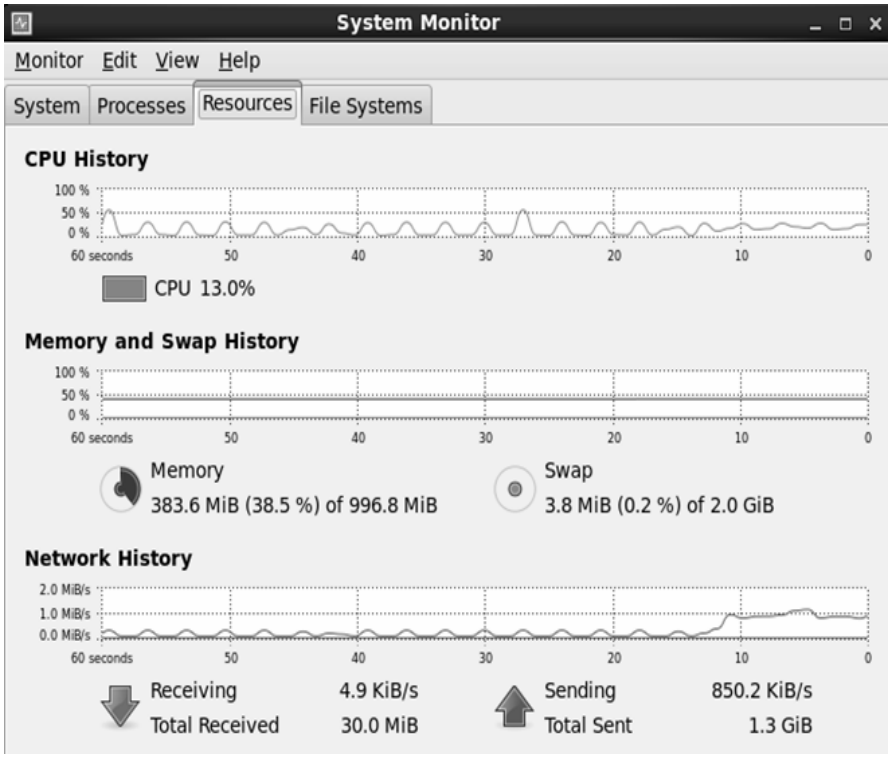
- Kan bruke (gode) GUI-verktøy for “System Monitoring”
- Fra shellet kan bl.a. disse kommandoene brukes:

`top`	Løpende systemstatus og informasjon om alle prosessene på systemet
`ps`	Skriv ut (deler av) prosesstabellen i Linux
`pstree*`	Skriver ut hele eller deler av “slektstreet” for prosessene

*: Se forelesningsnotater om “Prosesser og tråder”

GUI System Monitor





`top`: Dynamisk prosessinformasjon

- `top` viser:
  - Systemstatus øverst på skjermen
  - Liste med info om alle prosesser på systemet
  - Sortert på ressursbruk – “grådige” prosesser øverst
  - Listen oppdateres hvert tredje sekund (default)
- Fullskjerm – blokkerer kommandolinjen inntil exit
- Interaktiv – tastetrykk fra bruker forandrer virkemåten*

*: Trykk ‘`h`’ når `top` kjører for å se en oversikt over ulike opsjoner/kommandoer

Eksempel: Skjermbilde fra `top`

```
top - 08:18:12 up 9 days, 18:25,  3 users,  load average: 0.19, 0.05, 0.01
Tasks: 156 total,   1 running, 155 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.3%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1020756k total,   830872k used,   189884k free,     7868k buffers
Swap:  2064376k total,     4060k used,  2060316k free,   409576k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
  253 root      20   0     0    0    0 S  0.3  0.0   0:18.67 scsi_eh_1
16920 Student   20   0  292m  12m 9236 S  0.3  1.3   0:03.36 gnome-terminal
20536 Student   20   0 15088 1232  908 R  0.3  0.1   2:17.55 top
    1 root      20   0 19396 1332 1100 S  0.0  0.1   0:03.34 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.05 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.01 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    7 root      20   0     0    0    0 S  0.0  0.0   0:00.02 events/0
    8 root      20   0     0    0    0 S  0.0  0.0   0:00.00 cpuset
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.09 khelper
   10 root      20   0     0    0    0 S  0.0  0.0   0:00.00 netns
   11 root      20   0     0    0    0 S  0.0  0.0   0:00.00 async/mgr
   12 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pm
   13 root      20   0     0    0    0 S  0.0  0.0   0:00.00 sync_supers
   14 root      20   0     0    0    0 S  0.0  0.0   0:00.00 bdi-default
```

Tastetrykk som endrer virkemåte for `top`

A	Use alternate display mode
D	Alter interval time
L	Turn on/off load statistics
T	Turn on/off task statistics
M	Turn on/off memory statistics
f, o	Add/remove fields or alter display order
H	Show threads
U	Show specified user owned processes only
N	Show specific number of processes only
Q	Quit
I	Turn on/off including idle processes

`ps` – skriver ut prosesstabellen

- `ps` gir et “snapshot” av prosessene “akkurat nå”
- Svært mange opsjoner* for å spesifisere:
  - Hvilke prosesser som skal vises
  - Hva slags informasjon som skal vises
- `ps` er spesiell mht. opsjoner (historisk begrunnet):
  - Opsjoner som begynner med `-`
  - Opsjoner som begynner med `--`
  - Opsjoner som *ikke* begynner med en bindestrek

Noen eksempler på `ps`-kommandoer

- `ps`
  - Gir kun informasjon om prosesser som kjører i nåværende shell av deg, innlogget bruker
  - Oftest bare `bash`, `ps` og evt. bakgrunnsprosesser
- `ps a`
  - Viser alle prosessene tilhørende dette terminal-vinduet, inkludert systemprosesser som eies av `root`
- `ps x`
  - Viser alle prosessene til innlogget bruker, inkludert GUI
- ps u
  - “user-oriented”, mer info. om CPU- og minnebruk. etc.

Noen av outputkolonnene fra ps

PID PPID	Prosess- og forelderprosess-indentifikatorer
UID GID	Prosessens eier og gruppetilhørighet
%CPU	CPU-bruk
%MEM	Minnebruk
VSZ	Forbruk av (virtuelt) minne (KB)
RSS/RSZ	Forbruk av fysisk RAM (KB)
TTY	Terminalvinduet (? hvis ingen TTY)
START	Når prosessen startet
TIME	Totalt forbruk av CPU-tid
COMMAND	Kommando som startet prosessen
NI	Prioritet (niceness value)
STAT	Prosess-status, en tegnkode, se neste lysark →

`ps` – koder for prosess-status (**`STAT`**)

`D`	uninterruptible sleep (usually IO)
`R`	running or runnable (on run queue)
`S`	interruptible sleep (waiting for an event to complete)
`T`	stopped
`W`	paging
`X`	dead (should never be seen)
`Z`	"zombie" process, terminated, not killed by parent
`<`	high-priority (not nice to other users)
`N`	low-priority (nice to other users)

`nice` – prioritering av prosesser i Linux

- Prosesser får en “nice-verdi” (default 0) ved oppstart
- Nice-verdier er heltall med verdier fra -20 til 19

  19 : “Super-nice” prosess, prioriteres lavt, langsom

  -20 : “Grådig” prosess, prioriteres på topp, rask
- Brukere kan sette positiv nice-verdi for en prosess:

  `nice -15 `*`program`*
- Bare sys.adm./root kan bruke negative nice-verdier
- Prioritet til kjørende prosess kan endres med `renice`

Stoppe/drepe prosesser

- Prosesser terminerer vanligvis når de kommer til slutten av programkoden under eksekvering
- En prosess kan avslutte “unormalt” pga. run-time feil
- OS legger da ofte igjen en fil med navn `core`
- Dette er en “core dump”, som inneholder minnet for prosessen slik det var da den terminerte unormalt
- Noen ganger kan prosesser henge eller oppføre seg unormalt, og må stoppes av bruker
- Ofte vil en `Ctrl-C` (interrupt) fra kommandolinjen stoppe forgrunnsprossessen i shellet

Stopping av barneprosesser

- Hvis en barneprosess terminerer uten at forelder er tilgjengelig (f.eks. i sleep-status):
  - Barnet terminerer, men tas ikke ut av systemet
  - Barn må først “rapportere” exit til forelder
  - Barn blir til en “zombie”-prosess inntil forelder våkner
- Hvis forelderprosessen dør mens barnet fortsatt kjører, får vi en “orphan” prosess
  - Feilsituasjon, ingen andre enn `init` får være “foreldreløs”
  - `init` (evt. forelders forelder) vil da vanligvis “adoptere” barneprosessen

Brukerkommandoer for å drepe prosesser

Tre ulike alternativer:

1. Prøv å terminere programmet normalt med en “exit”-kommando e.l.

2. Stop prosessen fra en “GUI process monitor”:
   - Menyvalg: “Stop process” / “Kill process”

3. Stopp prosessen ved å sende den et *signal* med kommandoen `kill`

Linux-prosesser og signaler

- Alle prosesser i Linux responderer på *signaler*
- Signaler er beskjeder fra OS-et om at en prosess skal avslutte eller endre “oppførsel” fordi “noe uventet har skjedd”
- Kommandoen `kill` kan brukes for å sende et signal fra OS til en prosess
- Signalene identifiseres med nummer eller navn, noen vanlige signaler er gitt på de neste to sidene →

Name	#	Description
SIGHUP	1	Hangup
SIGINT	2	Terminal interrupt
SIGQUIT	3	Terminal quit
SIGILL	4	Illegal instruction
SIGTRAP	5	Trace trap
SIGIOT	6	IOT Trap
SIGBUS	7	BUS error
SIGFPE	8	Floating point exception
SIGKILL	9	Kill (can't be caught or ignored)
SIGUSR1	10	User defined signal 1
SIGSEGV	11	Invalid memory segment access
SIGUSR2	12	User defined signal 2
SIGPIPE	13	Write on a pipe with no reader, broken pipe
SIGALRM	14	Alarm clock
SIGTERM	15	Termination

Name	#	Description
SIGSTKFLT	16	Stack fault
SIGCHLD	17	Child process has stopped or exited, changed
SIGCONT	18	Continue executing, if stopped
SIGSTOP	19	Stop executing(can't be caught or ignored)
SIGTSTP	20	Terminal stop signal
SIGTTIN	21	Background process trying to read, from TTY
SIGTTOU	22	Background process trying to write, to TTY
SIGURG	23	Urgent condition on socket
SIGXCPU	24	CPU limit exceeded
SIGXFSZ	25	File size limit exceeded
SIGPWR	30	Power failure restart

`kill` – kommandoen

- `kill`: Send et signal til en prosess
- Brukere kan bare sende signaler til *egne* prosesser
- Liste over alle signaler: `kill -l`
- Spesielt signal: `SIGKILL` (9) for å stoppe prosess:
  - Sendes ikke videre til prosessen fra OS-et
  - Kalles en “sure kill”: Prosessen kan ikke selv stoppe eller håndtere en `SIGKILL`

Bruk av `kill`*

`kill PID`	Terminér en bestemt prosess
`kill -`*`signal`* `PID`	Send et gitt signal til en prosess
`kill -KILL PID`	"Sure kill"
`kill -KILL -1`	Massemord
`kill -HUP PID`	Brukes ofte for å *restarte* server-prosesser (web, ftp, o.l.)

*: Se også kommandoene `pkill` og `killall`

Avslutte og stenge ned Linix

- `shutdown time [`*`message`*`]`
  - Starter nedstenging av Linux til angitt tid
  - Stenger for nye login
  - Tid kan angis på mange måter, svært fleksibelt
- `shutdown` stopper bare Linux-kjernen
- Maskinen venter oftest på hva som skal gjøres videre:
  - halt – halt the system but leave the power on
  - poweroff – halt the system and turn the power off
  - reboot