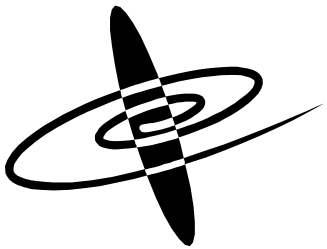
9 Processer

Processer



9 Processer

Hvad er en proces?

- En proces er et program under udførelse (eksekvering)
- Enhver bruger, der er logget ind, har mindst 1 proces kørende (fx. en shell)
- Enhver proces er knyttet til et bruger-ID, almindelig bruger eller systembruger (root, bin, adm o.s.v)
- Enhver proces er knyttet til et bruger-ID, men brugeren behøver *ikke* at være logget på:
  - Når systemet bootes startes der fx. processer, som er knyttet til brugerne root og bin. Disse processer lever videre i resten af den tid, UNIX/Linux-systemet er kørende (fx. init, scheduler, print-dæmoner m.fl.).
  - Baggrundsprocesser (bl.a. ved brug af ***nohup*** -beskrevet i modul 10)

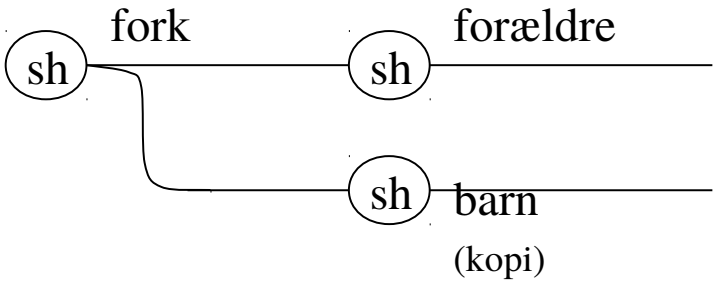
9 Processer

Hvordan behandles en proces?

- **En proces** kan af gode grunde ikke eksekvere hele tiden. Der er mange processer, men som skal deles om den CPU kraft der er til rådighed.
- Kernen sørger for, at de forskellige dele af en proces fordeles imellem RAM og swapfiler, for optimal behandling.
- Når en proces er kørende, kan der udføres 2 typer kode:
  - User code — Program skrevet af bruger eller kommando fra disken
  - Kernel code — Kode, som udføres i kernen efter kald fra user code – typisk i forbindelse med læsning fra eller skrivning til en angivet hardwareenhed (device).
- Denne adskillelse sikrer, at hele systemet ikke går ned, selv om en enkelt proces fejler.

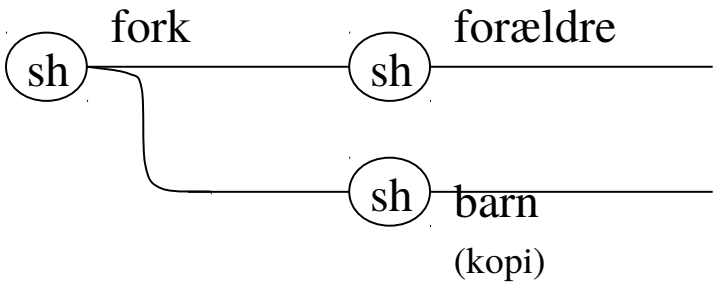
Hvordan udføres en ekstern kommando?

- Shell'en, som er det kørende program (proces), skriver `$` på skærmen.
- Brugeren indtaster en kommandolinje, som læses af shellen fra standardinput (stdin), f.eks.: *$ ls -l*
- Shell'en adskiller kommandoen og argumenterne.
- Shell'en søger i nogle få directories efter programmet (kommandoen), der skal udføres.
- Shell'en deler sig (forker) i to (næsten) identiske processer kaldet *forældre* og *barn*. Den laver en kopi af sig selv.



Hvordan udføres en ekstern kommando? (fortsat)

- Alle attributter fra forældren kopieres til barnet *(arbejdskatalog, environment variabler, åbne filer etc.)*.
- Begge processer kører nu parallelt.

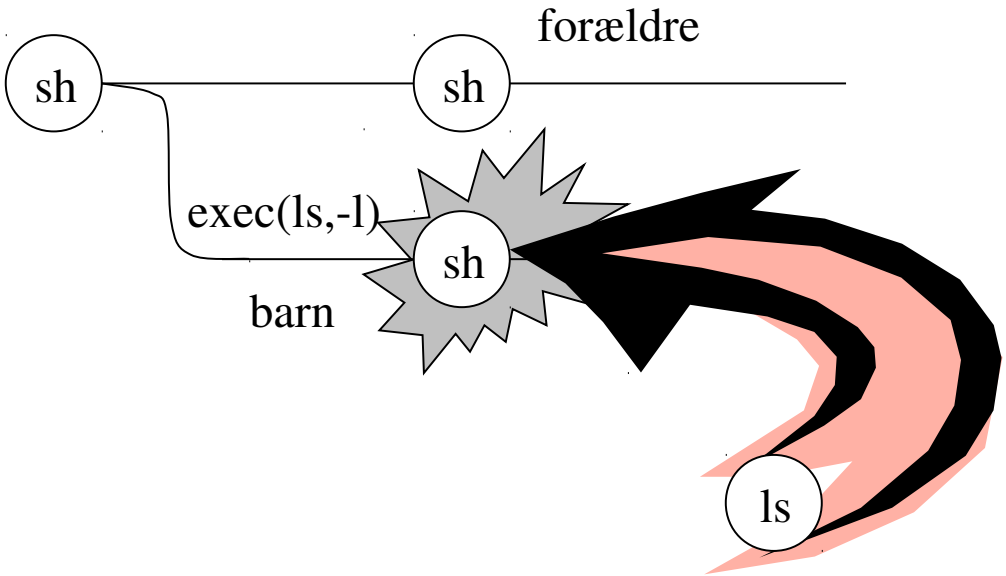


- Forældren venter nu på, at barnet skal afslutte. *(wait)*

(fortsættes)

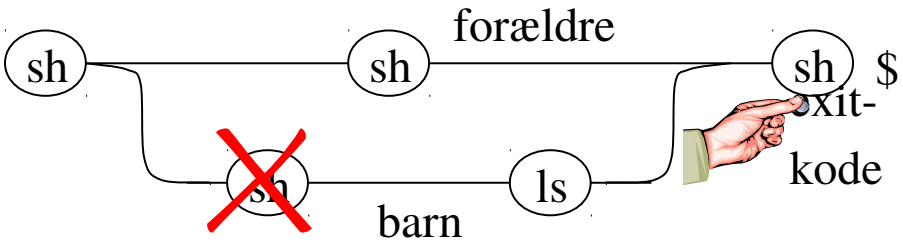
Hvordan udføres en ekstern kommando? (fortsat)

- Barnet, der startede sit liv som en shell, overskriver nu sig selv med et nyt program. Det bliver til det program, som brugeren kaldte med sin kommando (i vores eksempel **ls**.)



Hvordan udføres en ekstern kommando? (fortsat)

- Barneprocessen udfører brugerens opgave, afleverer exitværdi til forældreprocessen og afslutter derefter.



- Forældren (shell'en) vågner nu op og kører videre, d.v.s. viser brugeren en ny prompt. **$**

Fordele ved denne fremgangsmåde:

- Lille shell (den skal ikke kunne alting selv)
- Mulighed for at starte baggrundsprocesser (sideløbende)
- Shell-variable i forældreshellen er beskyttet mod ændringer.

9 Processer

Processer lige nu *(ps kommandoen)*:

- **Med kommandoen ps kan man få en oversigt over, hvilke processer der i øjeblikket findes på systemet**
- Syntax: ps [flag]

```
$ ps                    # Se egne processer – (i aktvt vindue)
$ ps -ef                # Se alle systemets processer
$ ps -u peter           # Se hvilke processer brugeren peter kører
```

Forgrundsprocesser

- En *forgrundsproces* er en proces, der kører i det aktuelle vindue.
- Dette *kan* indebære mere eller mindre ventetid afhængig af, hvor lang tid programmet er om at udføre sit arbejde og derefter afslutte.
- Måske er man inde at arbejde interaktivt i ***vi*** eller ***nano*** der hver har sit eget kommandosprog.
- Man får først sin *prompt* ($) tilbage, når forgrundsprocessen er afsluttet.

Baggrundsprocesser

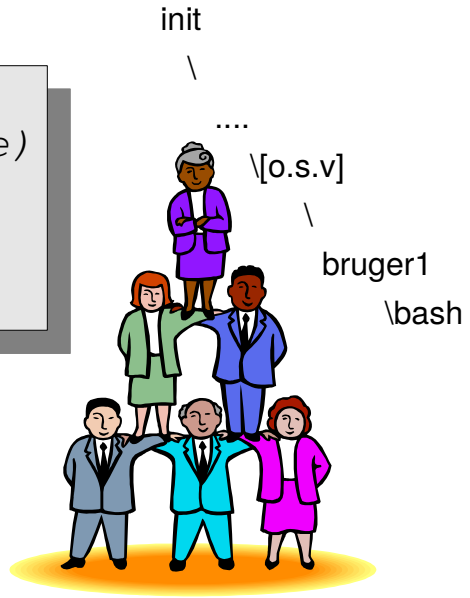
- En *baggrundsproces* er en proces, der kører i baggrunden – dvs. sideløbende med at man indtaster og udfører andre kommando(er).
  Dette giver mulighed for større effektivitet.
  Jævnfør UNIX/Linux-filosofien: Tænk parallelt.
- Når baggrundsprocessen startes, får man hurtigt sin prompt tilbage, så man kan indtaste flere kommandoer.

Mere om baggrundsprocesser i modul 10 *(Shellen og dens funktion).*

Proceshierarki

- Et *proceshierarki* er en struktur, der kan sammenlignes med et filtræ eller et stamtræ.
- Enhver proces har en forældreproces, der igen har en forældreproces, der igen…
- Man kan følge dette proceshieraki, når man bruger kommandoerne:

```
$ ps -ef |head -20    #(Se PID- og PPID-kolonnerne)
$ ps -axf             #(Semigrafisk visning)
$ pstree              #(Viser semigrafisk træstruktur)
```



SSH – log på en anden maskine

- **Med kommandoen *ssh* kan du logge på en anden maskine på netværket**

  SYNTAX: ssh [brugernavn]@[host(ip-adresse)]

  Eksempel:

```
$ ssh bruger1@192.168.[x].10[x]
        #(logger på instruktør maskinen )
```

Lav øvelser Modul 9.1

OBS! *I forbindelse med opgave 9.1.d skal du logge på instruktør maskinen*

NICE og RENICE

NICE

- Syntax: `nice [flag] [kommando]`
- En proces i UNIX/Linux har altid en prioritet,
  som standard er den ***0*** – så alle jobs prioriteres ens
- Man kan ændre denne ***prioritet*** på en ***skala fra -20 til 19***
  hvor *minus* ***-20*** er ***højest prioritet*** (”not nice”) og
  hvor *plus* ***19*** er ***lavest prioritet*** (”very nice”)

RENICE

- Syntax: `renice[-n][(+/-)prioritet][-g|-p|-u] ident.`
- Prioriteten af en allerede igangværende proces kan ændres ved hjælp af ***renice*** kommandoen.
- Det er kun ***root*** brugeren, der kan sætte en højere prioritet

NICE og RENICE

```
# Eksempel på stortjob.sh oprettes før demo:
$ vi stortjob.sh
```

```
#!/bin/bash
#job der gentager ls kommandoen i 4 minutter
end=$((SECONDS+240))
while [ $SECONDS -lt $end ]; do
ls >/dev/null
done
```

```
$ ./stortjob.sh &                    #(almindelig jobstart: nice værdi 0)
$ nice ./stortjob.sh & #(Starter job med default en nice værdi 10)
$ nice -11 ./stortjob.sh &           #(Starter og giver en nice værdi 11)
$ sudo nice --19 ./stortjob.sh &     #(niceværdi 20 = 1 prioritet)

Åbn nyt terminalvindue skriv
$ top                                #(Viser kørende processer)
$ renice -n -10 -p 3534 #(Gør proces nr. 3534 mindre "nice")
        #renice: failed to set priority for 3534 (process ID)
$ sudo renice -n -10 -p 3534 #(Gør proces nr. 3534 mindre "nice")
```