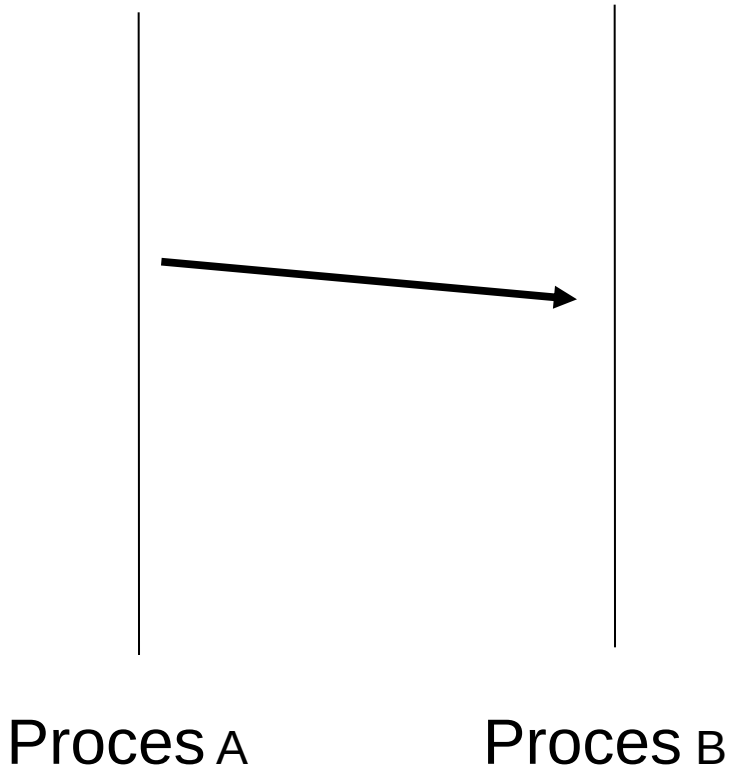
Signalen

Veerle Ongenae

Signalen

- Communicatie tussen processen
- Geen data
  - Nummer
- Asynchroon
  - Op elk moment
- Reactie
  - Standaardactie
  - Negeren
  - Opvangen



Signalen

- Sturen
- Ontvangen
- Kindproces volgen

Signalen sturen

- Gezonden van proces naar proces, veelal door kernel
- Voorbeeld: SIGSEGV (=segmentation violation)
  - wanneer een proces een ongeldig geheugenadres benadert
- Signaal dispositie (=reactie): negeren, opvangen, standaard actie uitvoeren (bijna altijd stoppen)

Signalen Sturen

- gebeurt met de functie kill

int **kill**(pid_t pid, int sig)

Return 0 bij succes, -1 bij fout

- pid
  - PID proces waarnaar gestuurd wordt
- sig
  - identificatie signaal

Overzicht signalen

- Veel gebruikte signalen :
  - SIGTERM/SIGKILL (einde, kill --> kan niet opgevangen worden)
  - SIGINT (onderbroken)
  - SIGCHLD (kind afgesloten)
  - SIGIO (IO mogelijk)
  - SIGALRM (alarmklok)
- Lijst
  - man signal

Sturen

- Wat doet deze code?

```
kill( getpid(), SIGTERM);
```

- Voorbeeld
  - proces_met_signaal.cpp

Signalen

- Sturen
- Ontvangen
- Kindproces volgen

Signaal Ontvangen

- alle signalen kunnen worden opgevangen en op eigen manier afgehandeld, behalve SIGSTOP en SIGKILL.
- signaal opvangen
  - een signal handler functie
  - Activeren
    - de functie sigaction

Signaal Ontvangen

```
#include <signal.h>
int sigaction(
  int signum,
  const struct sigaction *act,
  struct sigaction *old_act
)
```

return: 0 bij succes, -1 bij fout

Signaal Ontvangen

- signum: op te vangen signaal
- act: uit te voeren actie
- old_act: voorheen ingestelde actie

struct sigaction

```
struct sigaction {
 void (*sa_handler)(int);
 void (*sa_sigaction)(int, siginfo_t *, void *);
 sigset_t sa_mask;
 int sa_flags;
 void (*sa_restorer)(void);
}
```

Signaal Ontvangen

- in de struct sigaction hebben we enkel het veld sa_handler nodig; hierin komt de dispositie
  - SIG_DFL : default
  - SIG_IGN : negeer
  - een pointer naar een functie
    - Één parameter
      - Nummer signaal

Signaal Opvangen

```
void sig_functie(int signum){
  cout << “signaal ” << signum << endl;
}
int main(){
  struct sigaction sa;
  sa.sa_handler = &sig_functie;
  sigaction(SIGINT, &sa, NULL);
  ...
}
```

Voorbeeld

- signalen_opvangen.cpp
- Twee signalen
  - SIGUSR1
    - Aantal keer tellen
  - SIGTERM
    - Afsluiten + aantal uitschrijven
- Signalen sturen (shell)
  - kill –s SIG... PID
- ps (process snapshot)

Signalen

- Sturen
- Ontvangen
- Kindproces volgen

Kindprocessen volgen

- kind stuurt SIG_CHLD naar ouder wanneer het stopt
- ouder moet dit signaal opvangen en waitpid aanroepen in de handler

Voorbeeld

- volg_kinderen.cpp
- 5 kinderen starten
  - Slapen 3 sec
- SIGCHLD opvangen
  - Kindproces afsluiten

signalen blokkeren

- signalen komen niet op een queue maar zijn pending of niet