Concurrent servers met kindprocessen (fork)

Veerle Ongenae

Proces

- Instantie programma
  - Status
  - Werkmap
  - Toegangsrechten
  - Bestanden
  - Bronnen (geheugen, CPU rechten, ...)
- Verschillende processen simultaan
- ps: actieve processen

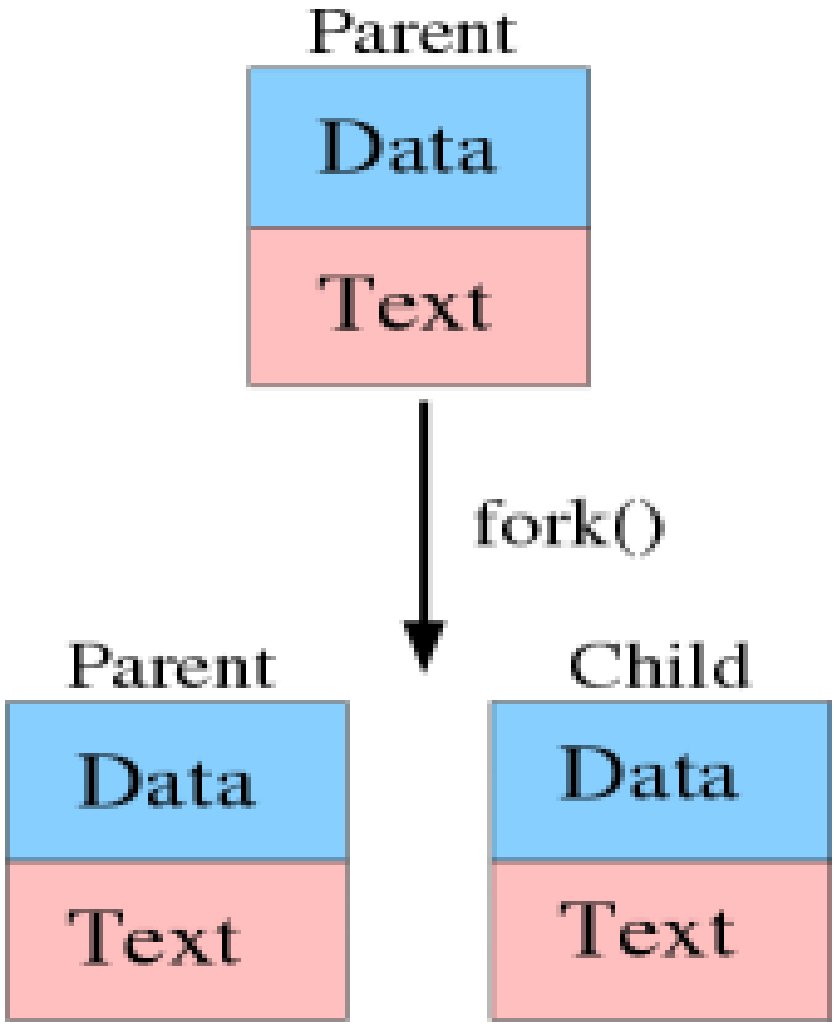
Eigenschappen proces

- pid: unieke identificatie
  - getpid
- ppid: pid ouder
  - getppid
- User ID en group ID
  - getuid, getgid
- CPU-tijd

Voorbeeld

- ToonProcesEigenschappen.cpp

Nieuw proces maken



een kind

```
int waarde;
waarde = fork();
cout << "waarde = " << waarde << endl;
```

- Uitvoer: /* volgorde kan omgekeerd!*/

```
waarde = 1536 // ouderproces
waarde = 0    // kindproces
```

fork()

- identieke kopie, behalve waarde van fork
- volgorde onbepaald
- alle variabelen gedupliceerd

```
#include <unistd.h>
pid_t fork(void)
```

Return: procesid (>0) in ouder, 0 in kind, -1 bij fout

Wachten op een kindproces

- Een ouder moet wachten op een kindproces
  - Anders kindproces = zombie
- Functies

#include <sys/types.h>

#include <sys/wait.h>

pid_t **wait**(int *status);

pid_t **waitpid**(pid_t pid, int *status, int options);

Return: PID, -1 bij fout

wait, waitpid

- status: exitstatus proces
- pid: PID van het proces waarop gewacht wordt
- options: opties
  - WNOHANG: niet blokkeren indien geen afsluitend proces
  - ...

exit status interpreteren

- Een aantal macro's
  - WEXITSTATUS
    - Waarde exitcode (argument exit, return)
  - WIFEXITED
    - Normaal beëindigd?
    - Beëindigd door niet afgehandeld signaal
  - WTERMSIG
    - Signaalnummer waardoor proces afgesloten werd

File descriptor bij kindprocessen

- Opstarten kindproces
  - Kopie descriptor
  - Kernel: teller voor elke descriptor
- Oproepen close
  - Teller één verminderen
  - Descriptor pas afgesloten indien teller = 0

Processen in netwerkprogrammatie

- Voor elke binnenkomende connectie nieuw proces --> gelijktijdig afhandelen clients
- Nog kunnen lezen van toetsenbord
  - Proces: communicatie shell
  - Proces: communicatie met client

Voorbeeld

- TCP_echo_server2.cpp

concurrent server

```
listenfd = socket(...);
bind(listenfd, ...);
listen(listenfd, ...);
while(1){
  connfd = accept(listenfd, ...);
  pid = fork();
  if (pid == 0) { // kind
    close(listenfd);
    verwerk_klant(connfd);
    close(connfd);
    exit(0);
  } else // ouder
    close (connfd);
}
```

Descriptor referentie teller

- close → sluit socket binnen het proces
- pas als d.r.c. == 0 is TCP verbinding gesloten

Voorbeeld

- Processen afsluiten ???
- TCP_echo_server3.cpp
- Beter alternatief
  - Signalen
  - Timer