Multiplexing met *select()*

Veerle Ongenae

IO Multiplexing

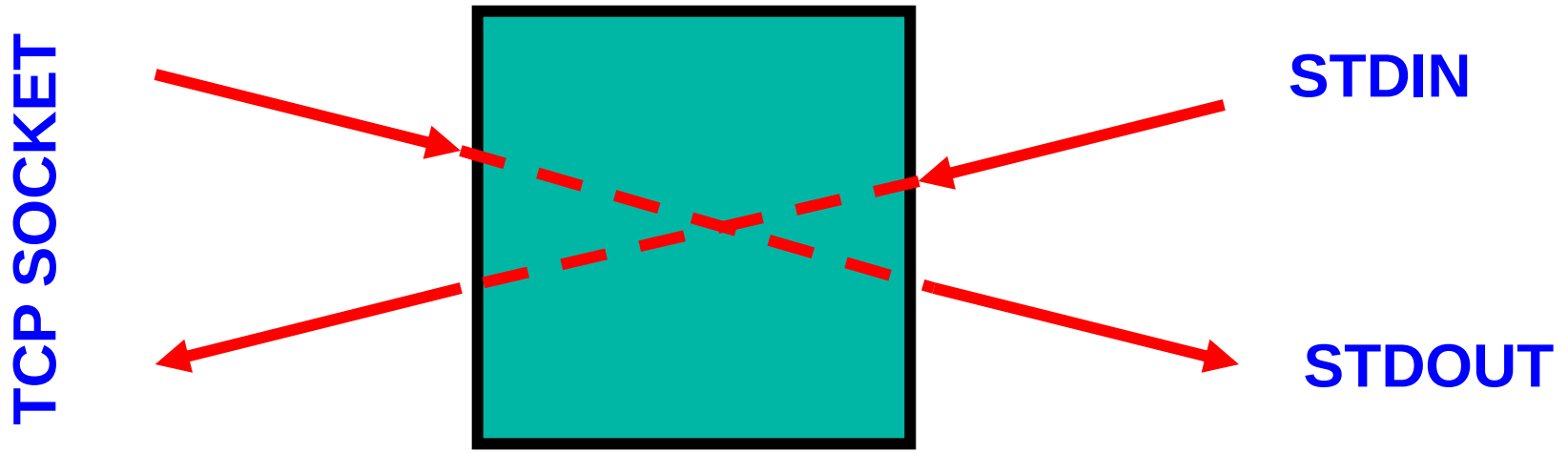
- Dikwijls willen we binnen een proces meerdere descriptoren gebruiken :
  - Een TCP client als telnet
    - Communicatie met toetsenbord
    - Communicatie met server
  - Wanneer we onverwachte situaties willen opvangen
  - Een server die zowel TCP als UDP aanvragen behandelt

Voorbeeld : een TCP client

- Input van het toetsenbord moet naar de socket gestuurd
- Input van de socket moet naar het scherm gestuurd

- Hoe weten we wanneer we moeten lezen van welke socket ??
  - Lezen blokkeert ...

Generische TCP Client



Opties

- Nonblocking IO (hfdst 22)
- Verschillende processen of threads (hfdst 15 en 20)
- Signalen en alarm (hfdst 16)
- Gebruik een functie die toelaat meerdere descriptoren tegelijk in het oog te houden : select()

select

```
#include <sys/select.h>
#include <sys/time.h>

int select (
        int maxfdp1,
        fd_set *readset,
        fd_set *writeset,
        fd_set *exceptset,
    const int struct timeval *timeout);
```

Return : aantal descriptoren die klaar zijn, 0 als timeout, -1 bij fout.

Parameters select

- maxfdp : aantal descriptoren dat moet worden getest, beginnend bij 0
  - Waarde hoogste descriptor + 1
- readset : set descriptoren waarvan we willen lezen
- writeset : set descriptoren naar waar we willen schrijven
- exceptset : set descriptoren met mogelijke excepties
- timeout : max tijd die select moet wachten

struct timeval

```
struct timeval {
   long tv_sec;     /* seconds */
   long tv_usec;    /* microseconds */
}

struct timeval max = {1,0};
```

select keert terug als :

- kan worden gelezen of geschreven
- uitzondering is opgetreden
- timeout-periode verstreken

- return waarde : aantal descriptoren waarmee we iets kunnen doen

fdset

- Specifiëren verzameling descriptoren
- macro's die je kan gebruiken

```
// set legen
void FD_ZERO(fd_set *fdset);

// descriptor toevoegen
void FD_SET(int fd, fd_set *fdset)

// descriptor verwijderen
void FD_CLR(int fd, fd_set *fdset)

// testen of descriptor in set zit
bool FD_ISSET(int fd, fd_set *fdset)
```

gebruik van select

- Maak fd_set
- Ledig hele set met FD_ZERO
- Voeg de descriptoren toe die je wil monitoren met FD_SET.
- select()
- Nadat select terugkeert, gebruik FD_ISSET om te checken of IO mogelijk is voor elke descriptor

```
  fd_set rset_all;
FD_ZERO(&rset_all); // leeg maken
FD_SET(invoer_fd, &rset_all); // invoer toevoegen
FD_SET(sockfd, &rset_all); // socket toevoegen

bool einde = false;
while (!einde) {
  fd_set rset = rset_all;
  maxfdp1 = max(invoer_fd, sockfd) + 1;

  select(maxfdp1, &rset, NULL, NULL, NULL);

  if (FD_ISSET(sockfd, &rset)) {// socket leesbaar?
    if (recv(sockfd, recvline, MAXLINE, 0) == 0)
      einde=true;
    else
      cout << recvline;
  }
```

```
/* invoer leesbaar ? */
if (FD_ISSET(invoer_fd, &rset)){
    if (read(invoer_fd, sendline, MAXLINE) == 0)
        FD_CLR(invoer_fd, &rset_all); /* klaar */
    else
        send(sockfd, sendline, strlen(sendline), 0);
}
} // while
```

shutdown

- host sluit connectie met close : zowel zenden als ontvangen onmogelijk
  - bv. echoclient wil verbinding verbreken na het sturen van de pakketten --> server kan niet meer lezen en terugsturen
- manier om zenden te sluiten (einde connectie) maar lezen open te houden : shutdown

shutdown()

```
#include <sys/socket.h>
int shutdown(int sockfd, int howto);
```

- SHUT_RD : sluit de connectie voor lezen, niet voor schrijven
- SHUT_WR : sluit de connectie voor schrijven, nietvoor lezen
- SHUT_RDWR : sluit de connectie volledig

```
  fd_set rset_all;
  FD_ZERO(&rset_all);
  FD_SET(invoer_fd, &rset_all);
  FD_SET(sockfd, &rset_all);
  bool einde = false;

  while (!einde) {
    fd_set rset = rset_all;
    maxfdp1 = max(invoer_fd, sockfd) + 1;

    select(maxfdp1, &rset, NULL, NULL, NULL);

      /* socket leesbaar? */
    if (FD_ISSET(sockfd, &rset)) {
      if (recv(sockfd, recvline, MAXLINE, 0) == 0)
        einde = true;
      else
        cout << recvline;
    }
}
```

```
/* invoer leesbaar */
if (FD_ISSET(invoer_fd, &rset)){
      if (read(invoer_fd, sendline, MAXLINE) == 0) {
        shutdown(sockfd,SHUT_WR);
        FD_CLR(invoer_fd, &rset_all); /* klaar */
      } else
        send(sockfd, sendline, strlen(sendline), 0);
    }
  }
```