UNIVERSITÀ DEGLI STUDI DI PARMA
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica-Informatica-Telecomunicazioni
a.a. 2001-2002

CORSO DI SISTEMI OPERATIVI A - ESERCITAZIONE 7

```
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

int  select(int n, fd_set *readfds, fd_set *writefds,
            fd_set *exceptfds, struct timeval *timeout);
```

La primitiva select attende il cambiamento di stato di un numero specificato di descrittori di file. Tre diversi insiemi di descrittori vengono controllati:

1. I descrittori contenuti in *readfds* vengono controllati per verificare la possibilità di eseguire una operazione di read non bloccante.
2. I descrittori contenuti in *writefds* vengono controllati per verificare la possibilità di eseguire una operazione di write non bloccante.
3. I descrittori contenuti in *exceptfds* vengono controllati per verificare eventuali eccezioni.

Il parametro *timeout* contiene il tempo di attesa della select. Se tale parametro viene fissato a zero la select ritorna immediatamente.
In uscita la select modifica gli insiemi per indicare quali descrittori hanno cambiato il loro stato. La primitiva ritorna il numero di descrittori il cui stato é cambiato.
Esistono alcune macro che consentono di manipolare gli insiemi dei descrittori dei file.

- `FD_ZERO(fd_set *set)`
  cancella un insieme;
- `FD_SET(int fd, fd_set *set)`
  aggiunge un descrittore ad un insieme;
- `FD_CLR(int fd, fd_set *set)`
  rimuove un descrittore da un insieme;
- `FD_ISSET(int fd, fd_set *set)`
  controlla se un descrittore appartiene ad un insieme;

1 Esercizi Proposti

Esercizio 1:

Si progetti in ambiente Unix/C la seguente interazione tra il processo server Ps e i suoi clienti Pi:

1. il processo Ps viene eseguito sul server darkstar;
2. Ps offre due servizi differenti su due porte il cui valore viene generato casualmente (nel caso in cui vengano generati numeri di porta gia' in uso, ripetere l'operazione); Ps stampa su stdout i valori generati;
3. i processi clienti Pi (anch'essi vengono eseguiti su darkstar) inviano richieste di servizio al server Ps attraverso socket di tipo Stream;
4. il processo Ps si avvale della primitiva select ponendosi in attesa delle richieste di servizio (con un timeout fissato di 5 minuti oltre il quale il server termina la propria attività);
5. il primo servizio offerto da Ps prevede la restituzione della data e dall'ora correnti ai processi clienti Pi;
6. il secondo servizio offerto da Ps prevede la restituzione ai processi client Pi di informazioni di rete (DNS) su un particolare host. Il nome dell'host viene inviato a Ps dagli stessi clienti.
7. Per soddisfare il secondo tipo di servizi Ps esegue le seguenti operazioni:
   - attiva un processo figlio per ogni nuova richiesta di servizio accettata;
   - il processo figlio esegue il comando *nslookup <nome host>*
     [si consiglia l'utilizzo della primitiva *system.* Esempio: *system(stringa);* dove *stringa = "nslookup darkstar"*];
   - il processo figlio salva l'output del comando precedente su un file;
   - il processo figlio in risposta al cliente scrive sulla socket il contenuto del file;
8. I processi Pi utilizzano il comando *telnet darkstar <porta>* per connettersi al server Ps (nel caso di richiesta del secondo tipo di servizio, i processi Pi inviano a Ps la stringa <nome host> dallo shell in seguito al comando *telnet*)

Di seguito viene riportata una traccia per l'esecuzione dell'attesa bloccante del server su una socket (FD_SETSIZE è il massimo numero di descrittori che possono essere contenuti all'interno di insiemi del tipo fd_set).

```
.....
struct timeval timeRec;
int sock1, sock2, rc;
fd_set select_set;

timeRec.tv_sec = 60*5;
timeRec.tv_usec = 0;
FD_ZERO(&select_set);
FD_SET(sock1, &select_set);
FD_SET(sock2, &select_set);

/* Test di lettura sulla socket */
rc = select(FD_SETSIZE, &select_set, 0, 0, &timeRec);
......[controllo timeout]......
if ((rc > 0) && FD_ISSET(sock1, &select_set))
......
```

Esercizio 2:

Il sorgente riportato di seguito (e presente nel direttorio */home/soa/eserc7*) è relativo a un semplice server HTTP.
Compilare e porre il server in esecuzione su darkstar, passando come argomento il numero di porta su cui deve mettersi in ascolto.
Lanciare il browser Netscape da Windows.
Scrivere nel campo Location del browser la stringa *172.28.14.253:<numero_porta>* per effettuare una connessione con il server; automaticamente viene ricevuta e visualizzata una pagina HTML con il contenuto del direttorio in cui si trova l'eseguibile del server.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>

#define BUFSIZE 1024


int main(int argc, char *argv[])
{
  int sock, nuovasock;
  struct sockaddr_in nomesocket;
  char buf[BUFSIZE];
  FILE *lfp;

  /* Controllo degli argomenti */
  if (argc != 2)
    {
      printf("Uso: %s <numero porta>\n", argv[0]);
      exit(1);
    }

  /* Crea la socket */
  sock = socket(AF_INET, SOCK_STREAM, 0);
  if (sock < 0)
    {
      perror("creazione stream socket");
      exit(2);
    }

  /* Prepara le informazioni per il binding */
  nomesocket.sin_family = AF_INET;
  nomesocket.sin_addr.s_addr = INADDR_ANY;
  nomesocket.sin_port = htons(atoi(argv[1]));

  /* Collega la socket alla porta specificata */
  if (bind(sock, (struct sockaddr *) &nomesocket, sizeof(nomesocket)) < 0)
```

```
  {
    printf("bind su stream socket");
    exit(3);
  }

/* Ascolta le richieste di connessione sulla porta */
if (listen(sock, 1) < 0)
  {
    perror("listen error");
    exit(4);
  }
do
  {
    /* Accetta una delle connessioni pendenti */
    nuovasock = accept(sock, 0, 0);

    /* Mostra i messaggi che riceve dal client*/
    close(0);
    dup(nuovasock);
    while(buf[0]!= 0x0d && buf[0]!=0x0a )
      {
        fgets(buf,128,stdin);
        fprintf(stderr,"%s",buf);
      }

    /* Ricava la lista dei file contenuti nel direttorio corrente*/
    system("ls > lista");
    /* Apre listfile in lettura */
    lfp = fopen("lista", "r");

    /* Crea e invia pagina HTML */
    strcpy(buf, "<html><head><title> Lista </title></head>\n");
    strcat(buf,
           "<body>\n<h2>File contenuti nel direttorio del server:</h2>\n<ul>\n");
    write(nuovasock, buf, strlen(buf));
    while (fgets(buf, BUFSIZE, lfp) != NULL)
      {
        write(nuovasock, "<li>", 4);
        write(nuovasock, buf, strlen(buf));
      }
    fclose(lfp);
    write(nuovasock, "</ul>\n", 7);
    strcpy(buf,"</body>\n</html>\n\n");
    write(nuovasock, buf, strlen(buf));

    /* Chiude il canale di comunicazione con il client*/
    shutdown(nuovasock,1);
    close(nuovasock);
```

```
    }
  while(1);
}
```

Si noti l'utilizzo delle funzioni di alto livello fopen(), fgets(), fclose() rispettivamente per l'apertura, lettura una riga alla volta e chiusura del file di testo chiamato "lista".

Esercizio proposto:

Realizzare la versione con socket Datagram dell'Esercizio 1 (attenzione: Telnet non funziona con il protocollo UDP, per cui bisogna realizzare un client "UDP-Telnet" semplificato).