

SISTEMI OPERATIVI

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 proprio host;
2. Ps offre due servizi differenti su due porte il cui valore viene specificato come argomento di invocazione del programma;
3. i processi clienti Pi (anch'essi vengono eseguiti sul proprio host) 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 di informazioni di rete (DNS) su un particolare host ai clienti. Il nome dell'host viene inviato dai processi clienti stessi a Ps.
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 host <porta>` per connettersi al server Ps (nel caso di richiesta del secondo tipo di servizi 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);
if (rc>0 && FD_ISSET(listenSocket,&select_set))
.....

```

Esercizio 2:

Si realizzi un **server concorrente** su socket **STREAM** che abbia il seguente comportamento:

- il server riceve dai clienti una stringa che rappresenta un intero N ($2 \leq N \leq 10$) ;
- il server deve quindi creare N processi figli che attendono un intervallo di durata casuale D_i ($1 \leq D_i \leq 5$) secondi, visualizzano un messaggio con il loro PID e terminano ;
- il server comunica al client il PID del processo figlio che è terminato per primo ;
- se il server riceve un segnale SIGUSR2 durante l'attesa dei processi figli, deve provvedere a terminarli tutti e ad inviare il messaggio "Tutti i processi figli sono stati terminati" al cliente.

Si richiede l'utilizzo della gestione affidabile dei segnali.

Si suggerisce l'utilizzo del comando *telnet* come client per la verifica del funzionamento del server.

Esercizio 3:

Si progetti in ambiente Unix/C, avvalendosi del supporto dei segnali affidabili, la seguente interazione di processi:

- il sistema consiste di due triple di processi ($\{P_A, P_{A1}, P_{A2}\}$ e $\{P_B, P_{B1}, P_{B2}\}$) in esecuzione su due diversi nodi A e B di una rete ;
- all'interno di ciascun nodo la comunicazione avviene attraverso una o più pipe ;
- i processi P_A e P_B comunicano attraverso una socket TCP alla porta 4001;
- inviando da shell un segnale SIGUSR1 al processo P_{A1} , lo stesso segnale deve essere inoltrato al processo P_{B1} , utilizzando una opportuna comunicazione attraverso le pipe e la socket;
- ricevuto il segnale SIGUSR1, il processo P_{B1} deve trasmettere (sempre utilizzando una opportuna comunicazione attraverso le pipe e la socket) a P_{A1} un messaggio contenente data/ora e il numero di segnali ricevuti.
- inviando da shell un segnale SIGUSR2 al processo P_{B2} , esso deve far giungere al processo P_{A2} , utilizzando una opportuna comunicazione attraverso le pipe e la socket, un messaggio contenente l'elenco degli utenti del nodo B.

Esercizio 4:

Nel direttorio /home/francesco.zanichelli/so-src-lab/eserc7 è presente il sorgente "serve-HTTP2.c" di un semplice server HTTP. Compilare e porre il server in esecuzione sul proprio PC, passando come argomento il numero di porta su cui deve mettersi in ascolto. Eseguire un browser e scrivere nel campo Location del browser la stringa nome_pc: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.