Attente simultanée sur plusieurs fd : SELECT

int **select (**int max1, fd_set *lecteurs, fd_set *ecrivains, fd_set *exceptions,
struct timeval *delai_max**)**
Attente simultanée sur trois ensembles de descripteurs,
max1 = numéro du plus grand descripteur + 1,
bloquant pendant delai_max,
retourne les descripteurs correspondant à une E/S

FD_ZERO (fd_set* ensemble**)**
Mise à zéro de l'ensemble
FD_SET (int fd**,** fd_set* ensemble**)**
Ajoute un descripteur à l'ensemble
FD_CLR (int fd**,** fd_set* ensemble**)**
Supprime un descripteur de l'ensemble
FD_ISSET (int fd**,** fd_set* ensemble**)**
Teste si un descripteur est dans l'ensemble

```
int main(int argc, char *argv[]){
  ….   /* initialisation  idem serveurTCP.c */
  printf("Appuyer sur une <Entree> pour tuer le serveur\n");
  /* Boucle principale */
  for (;;) {
     fd_set mselect;
   /* Construire le masque du select */
    FD_ZERO(&mselect);
    FD_SET(0,&mselect);  /* stdin */
    FD_SET(sc,&mselect); /* la socket */

    if (select(1024, &mselect, NULL, NULL, NULL) == -1) {
      perror("select");
      exit(3);
    }
/*** Un evenement a eu lieu : tester le descripteur ***/

    if (FD_ISSET(0,&mselect)) {
      /* Sur stdin */
      break; /* Sortie de la boucle */
    }
    if (FD_ISSET(sc, &mselect)) {
      /* Sur la socket de connexion */

      /* Etablir la connexion */
      if ( (scom = accept(sc, (struct sockaddr *)&exp,
&fromlen))== -1) {
```

```
            perror("accept");
            exit(2);
        }
            /*** Lire le message ***/
        if (read(scom,message, sizeof(message)) < 0) {
            perror("read");
            exit(1);
        }
           ….
           /* Fermer la connexion */
           shutdown(scom,2);
           close(scom);
        }

  } /* Fin de la boucle */

  close(sc);
  return 0;
}
```