TELECOM Nancy — UNIVERSITÉ DE LORRAINE

RSA -Réseaux
2013-2014
TD Programmation Avancée

1 Select

Exercice 1 Modifier la version du serveur echo dont vous avez un extrait ci -dessous pour en faire un serveur multi-clients mais sans le fork. La fonction str_echo(int sockfd) lit le message reçu depuis le descripteur sockfd et le renvoie sur ce même descripteur. Quand le client a fermé la connexion, str_echo() retourne 0.

```
  int sockfd, n, newsockfd, childpid, servlen,fin;
  struct sockaddr_in  serv_addr, cli_addr;
  socklen_t clilen;
....

if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) <0) {
   perror("servmulti_:_Probleme_socket\n");
   exit (2);
 }
....

 for (;;) {

   clilen = sizeof(cli_addr);
   newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr,  &clilen);
   if (newsockfd < 0) {
    perror("servmulti_:_erreur_accept\n");
    exit (1);
   }


  if ( (childpid = fork() ) < 0) {
    perror ("server:_fork_error\n");
    exit (1);
  }
  else
    if (childpid == 0) {
      close (sockfd);
      while (str_echo (newsockfd));
      exit (0);
    }

  close(newsockfd);
 }
}
```

1. Définir l'algorithme général ainsi que les différentes structures nécessaires pour réaliser ce serveur
2. Ecriture du code de manière plus précise sans forcément tout décrire. Vous testerez le code en TP.

Exercice 2 Soit un processus inetd qui attend des requêtes venant de plusieurs clients utilisant soit le protocole TCP ou le protocole UDP. L'ensemble des services à considérer ainsi que les exécutables associés sont définis dans un fichier (/etc/inetd.conf) :

service-name	endpoint-type	protocol	wait-status	uid	server-program	server-arguments
ftp	stream	tcp	nowait	root	/usr/bin/ftpd	ftpd
tftp	dgram	udp	wait	root	/usr/bin/tftpd	tftpd -s /tftpboot
telnet	stream	tcp	nowait	guest	/usr/sbin/telnetd	telnetd

Les numéros de port relatifs à ces services sont précisés dans /etc/services.
L'objectif de inetd est de faire qu'un seul processus attende ; ce processus a ensuite en charge de lancer l'exécutable associé au service demandé par le client lorsqu'une requête du client arrive.

1. Préciser, par un schéma, les primitives réseau/système que le serveur inetd devra appeler avant de se mettre en attente de requêtes clients.
2. Comment sera réalisée cette attente ?
3. Un client UDP ou TCP arrive. Illustrer par un schéma pour mettre en évidence l'algorithme effectué par le serveur.
4. Le flag nowait spécifie que le serveur inetd n'a pas besoin d'attendre que son fils se termine avant d'accepter une nouvelle connexion pour ce service. Pourquoi est-ce le cas avec des services utilisant TCP et non par avec un service comme tftp ?
5. Que faut-il ajouter pour dans le processus tftp pour permettre à plusieurs clients tftp de se connecter simultanément ?

2 Socket Raw

Exercice 3 Donner trois caractéristiques que permettent les socket raw et ne permettent pas les sockets UDP et TCP

Exercice 4 1. Que fait cet extrait de code suivant :

```
1  if ((serverSocket = socket(AF_INET, SOCK_DGRAM, 0)) <0) {
2    perror ("erreur_socket");
3    exit (1); }
4
5  ttl=1;
6  if (setsockopt(serverSocket, IPPROTO_IP, IP_TTL, &ttl, sizeof(ttl)) <0){
7    perror ("setsockopt");
8    exit (1);
9  }
10
11 ftime(&tp);
12 data =ctime(&tp.time);
13
14 if ( (n= sendto (serverSocket, data, strlen(data),0,
15      (struct sockaddr *)&serv_addr, sizeof(serv_addr)
16           )) != strlen(data))  {
17   perror ("erreur_sendto");
18   exit (1);
19 }
20
21 if ( (n= recvfrom (serverSocket, sendbuf, sizeof(sendbuf),0,
22   (struct sockaddr *)&serv_addr, &len)) != strlen(data) )  {
23   printf ("erreur_sendto");
24   exit (1);
25 }
26
27 sendbuf[n]='\0';
28 printf ("Message_recu_du_serveur_%s_\n", sendbuf);
29 close(serverSocket);
```

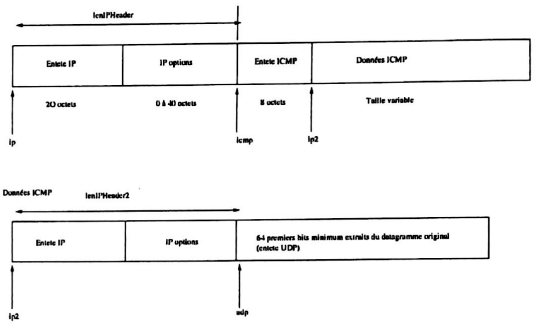
2. Si vous exécutez ce code sur votre machine, pouvez me donner les différents scénarios possibles

Exercice 5

Compléter le code ci-dessous pour écrire un daemon icmp signalant l'arrivée de messages ICMP de type ICMP_ECHO et ICMP_ECHOREPLY ICMP_UNREACH et ICMP_TIMXCEED.

```
main (argc, argv)
int argc;
char *argv[];

{
  int rawSocket, n,lenIPHeader, lenIPHeader2, sport, dport;
  struct sockaddr_in  fromAddr;
  socklen_t len;
  char source[BUFSIZE],recvbuffer[BUFSIZE];;
  struct ip *ip, *ip2;
  struct icmp *icmp;
  struct udphdr *udp;

  if ((rawSocket = socket(....., ....., .....)) <0) {
    perror ("erreur_socket");
    exit (1); }

  for (;;) {
    len = sizeof(fromAddr);
    if ( (n= recvfrom (....., ..... BUFSIZE,0,
                  (struct sockaddr *)&(fromAddr, &len)) < 0 )  {
      printf ("erreur_recvfrom");
      exit (1);}

    if ( inet_ntop (AF_INET, (const void *)&fromAddr.sin_addr,source ,len) <0) {
      printf ("erreur_inet_ntop");
      exit (1); }
    printf( "%d_octets_ICMP_de_%s:_\n", n, source);

    ip = .... ;                          // debut entete IP
    lenIPHeader = ip->ip_hl * .... ;     // ip->ip_hl longueur  en mot de 32 bits

    icmp =  ..... ;              // debut entete ICMP
    ip2 =  .....;                // debut en-tete IP contenu dans ICMP
    lenIPHeader2 = ..... ;       // longueur  en-tete IP
    if (ip2->ip_p == IPPROTO_UDP) {
      udp = ..... ;                      // debut en-tete UDP
      sport = ntohs(udp->uh_sport);
      dport = ntohs(udp->uh_dport) ;
      printf ("_port_source_=_%d_et_port_destination_=_%d_\n", sport, dport);
    }
    switch (icmp->icmp_type) {
    case ICMP_UNREACH: {
      printf ("destination_unreachable_\n");
      switch (icmp->icmp_code){
      case ICMP_UNREACH_PORT:
        printf ("_bad_port_\n");
        break;
      default:
        printf ("type_%d,_code_=_%d\n", icmp->icmp_type,
          icmp->icmp_code);
        break;
      } }
      break;
    case ICMP_ECHO:
      printf ("_echo_service__\n");
      break;
    case ICMP_ECHOREPLY :
      printf ("_echo_reply__\n");
      break;
    case ICMP_TIMXCEED :
      printf ("_Time_Exceed_\n");
      break;
    default:
      printf ("type_%d,_code_=_%d\n", icmp->icmp_type,
          icmp->icmp_code);
    }
}
```


Pour rappel le format d'un paquet ICMP est le suivant :



Exercice 6 Proposer une solution pour que l'application de l'exercice 4 puisse être informée que des messages ICMP sont arrivés suite à l'envoi des messages UDP. Vous considérez qu'il existe une fonction unix sendmsg/rcvmsg qui permet de passer des descripteurs de socket d'un processus à un autre via des sockets de type unix en mode STREAM.