Calcolatori Elettronici 2
5 Giugno 1998

Esercizio di programmazione distribuita in ambiente Unix

Si progetti un'applicazione distribuita Client/Server per una rete di workstation UNIX (BSD oppure System V). In particolare, il Client richiede al processo Server la stampa di un file su una stampante collegata alla macchina ove esegue il Server stesso.
Il Client presenta l'interfaccia:

esame nodoserver nomefile

dove ***nodoserver*** specifica l'indirizzo logico del nodo contenente il processo Server e ***nomefile*** è il nome del file che il Client invia al Server per ottenerne la stampa.
Nella progettazione del Server si consiglia di utilizzare il comando `lpr` per la stampa del file. In tale caso, si supponga che il comando `lpr` si incarichi della stampa di un file che deve però essere presente nel direttorio /var/spool/stampa. Si ipotizzi che `lpr` preveda come unico argomento il nome del file da stampare.

Si progetti sia il Client sia il Server concorrente.

```
/* compito del 5 giugno 1998 */
/* Client */

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

#define DIM_BUFF 512

void main( int argc , char *argv[] ) {
    int fd, sd , letti;
    char buff[DIM_BUFF];
    struct hostent *host;
    struct sockaddr_in rem_indirizzo;

    rem_indirizzo.sin_family= AF_INET;
    host=gethostbyname(argv[1]);
    if(host==NULL) {
        printf("host %s non trovato\n", argv[1]);
        exit(1);
    }
    rem_indirizzo.sin_addr.s_addr=((struct in_addr *)
                                   (host->h_addr))->s_addr;
    rem_indirizzo.sin_port= 55555;

    sd= socket(AF_INET, SOCK_STREAM, 0);
    if( sd<0 ) { perror("errore socket"); exit(1); }

    connect(sd, (struct sockaddr*) &rem_indirizzo,
                                sizeof(struct sockaddr_in));
    if( sd<0 ) { perror("errore connect"); exit(1); }

    fd= open(argv[2], O_RDONLY);
    if(fd<0) { perror("errore open"); exit(1); }

    while((letti=read(fd, buff, DIM_BUFF)) > 0 )
        write(sd, buff, letti);
    close(sd); close(fd);
    exit(0);
}
```

```
/* Server */

#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
#include <signal.h>

#define DIM_BUFF 512

void gestore(int signo) {
  int stato;
  wait(&stato);
}

main() {
  int fd, sd, newsd, letti, stato, ret;
  struct sockaddr_in indirizzo_server;
  char buff[DIM_BUFF], nomefile[30];

  sigset(SIGCHLD, gestore);

  memset((char *)&indirizzo_server, 0,
                          sizeof(struct sockaddr_in));

  sd=socket(AF_INET, SOCK_STREAM, 0);
  if(sd<0) { perror("errore socket"); exit(1); }

  indirizzo_server.sin_family=AF_INET;
  indirizzo_server.sin_port= 55555;

  ret=bind(sd, (struct sockaddr *) &indirizzo_server,
                          sizeof(struct sockaddr_in));
  if(ret<0) { perror("errore bind"); exit(1); }

  listen(sd, 10);
  chdir("/var/spool/stampa");
```

```
  for(;;) {
    newsd=accept(sd, 0, 0);
    if (newsd<0)
        if (errno==EINTR) continue;
            else exit(1);

    if(fork()==0) { /* figlio */
        close(sd);
        sprintf(nomefile,"file%d", getpid());
        fd=open(nomefile, O_WRONLY|O_CREAT, 0644);
        if(fd<0) { perror("errore open"); exit(1); }

        while( (letti=read(newsd, buff, DIM_BUFF))>0 )
            write(fd, buff, letti);

        close(fd);
        close(newsd);
        sigset(SIGCHLD,SIG_DFL);

        if(fork()==0) { /* nipote */
            execl("/bin/lpr", "lpr", nomefile, (char *) 0);
            exit(1);
        }
        wait(&stato);
        unlink(nomefile);
        exit(0);
    }
    close(newsd);
  } /* fine del for(;;) */
}
```