Sistemi di Elaborazione

Struttura dei server

Tipologie di server

- Server iterativo
  - serve solamente una richiesta alla volta
- Server concorrente:
  - Per ogni richiesta da parte di un client (accettata dalla *accept*) il server
    - Crea un processo figlio (primitiva fork()), oppure
    - Crea un thread, oppure
    - Attiva un thread da un pool creato in anticipo (pre-fork)
  - Il processo/thread gestisce la richiesta del client in questione

Server concorrente multiprocesso

```
for(;;) {
    consd = accept(listensd, …);          /* may block */
    if((pid = fork()) = = 0) {
        close(listensd);          /* child closes listening socket */
        doit(consd);              /* process the request */
        close(consd);             /* done with this client */
        exit(0);                  /* child terminates */
    }
    close(consd);                 /* parent closes conn. socket */
}
```

Server concorrente multiprocesso

```
#include <sys/types.h>
#include <unistd.h>
...
int sock, cl_sock, ret;
struct sockaddr_in srv_addr, cl_addr;
pid_t child_pid;
sock = socket(AF_INET, SOCK_STREAM,0);
if(sock==-1){ /*errore*/}
/*inizializzazione srv_addr*/
bind(sock, &srv_addr, sizeof(srv_addr));
listen(sock,QUEUE_SIZE);
for (;;) {
    cl_sock = accept(sock, &cl_addr, sizeof(cl_addr));
    if(cl_sock==-1){ /*errore*/}
    /* gestione cl_addr */
    child_pid = fork();
    if(child_pid==0) /* sono nel processo figlio*/
        gestisci_richiesta(cl_sock, sock, ...); /*funzione per la gestione delle
    richieste per il det servizio */
    else /* sono nel processo padre*/
        close(cl_sock);
}
```

Server concorrente multi-threaded

```
int consd;
int* par;
For(;;) {
        consd = accept(listensd, …);     /* probably blocks */
        par = malloc(sizeof(consd));
        *par = consd;
        if ( pthread_create( &tid, NULL, doit, (void*)par ) ) {
                        exit(0);                /* error on thread creation */
        }
}
```

- Chiusura della connessione
  - in questo caso la close(consd) viene fatta dal thread gestore (all'interno della funzione gestione_richiesta)

Server concorrente multi-threaded

```
#include <sys/types.h>
#include <unistd.h>
…
void* gestisci_richiesta( void* socket ) { /*funzione per la gestione delle richieste */ }
…
int sock, cl_sock, ret;
Int* par;
struct sockaddr_in srv_addr, cl_addr;
pthread_t tid;
sock = socket(PF_INET, SOCK_STREAM,0);
if(sock==-1){ /*errore*/}
/*inizializzazione srv_addr*/
bind(sock, &srv_addr, sizeof(srv_addr));
listen(sock,QUEUE_SIZE);
while(1){
	cl_sock = accept(sock, &cl_addr, sizeof(cl_addr));
	if(cl_sock==-1){ /*errore*/}
	/* gestione cl_addr */
	par = malloc(sizeof(cl_sock));
	*par = cl_sock;
	if ( pthread_create( &tid, NULL, gestisci_richiesta, (void*)par ) ) {
		exit(0);		/* errore */
	}
}
```