

2024년 1학기 시스템프로그래밍실습 10주차

FTP2-3

System Software Laboratory
Dept. of Computer Engineering,
Kwangwoon Univ.

Contents

▶ Requirements

- ▶ Socket
- ▶ Command Conversion
- ▶ Concurrent Server
- ▶ Signal Process
- ▶ Etc
- ▶ Sample Result

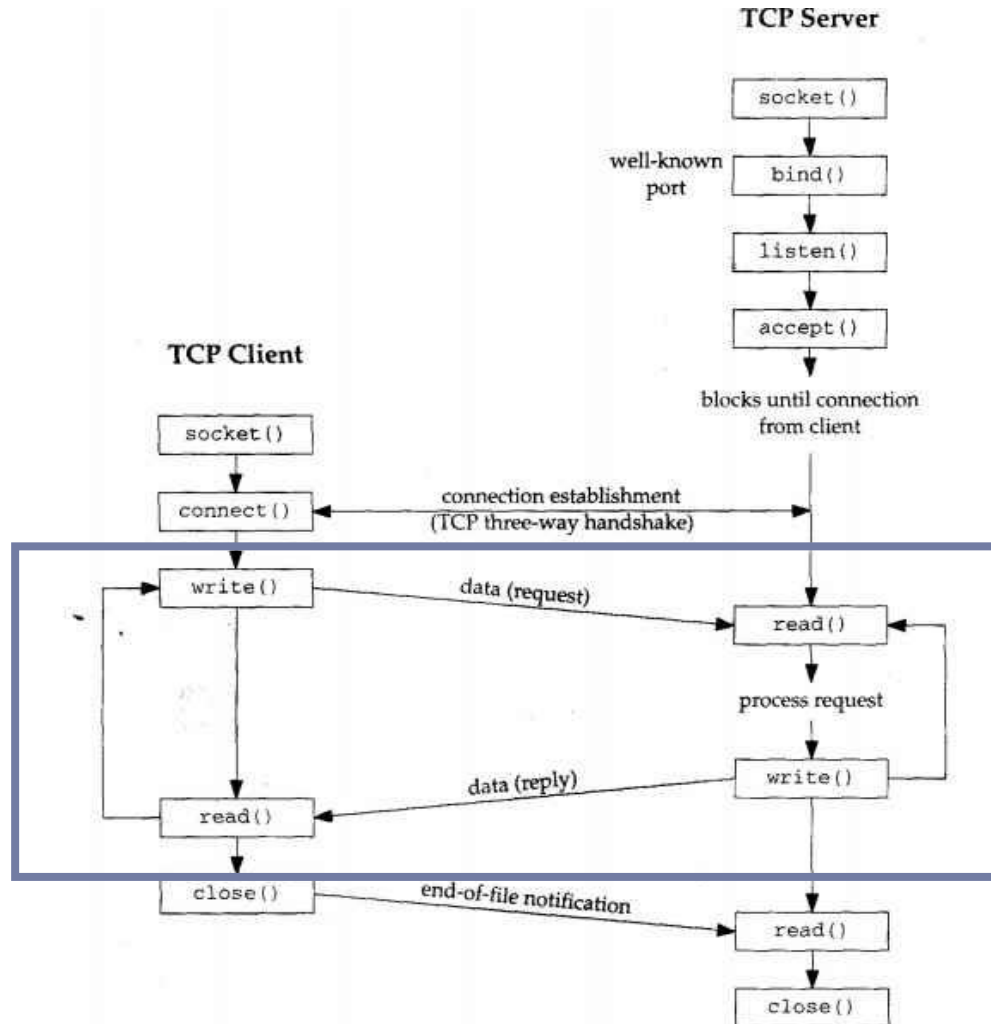
Schedule

- ▶ 1주 → Basic Socket Programming (with /s)
- ▶ 2주 → fork() and Signal Processing
- ▶ 3주 → Concurrent Server, Q & A

Requirement – Socket (1/4)

- ▶ **Assignment #1 + Socket algorithm**
 - ▶ Assignment 1 combined socket algorithm.
 - ▶ Simulate socket algorithm
 - ▶ Server : socket(), bind(), listen(), accept()
 - ▶ Client : socket(), connect()

Requirement – Socket (2/4)



Requirement – Socket (3/4)

▶ **Server side**

- ▶ When client connects, display client IP address , port number and child process ID
- ▶ Display client's commands.
- ▶ Check the FTP command and execute
- ▶ Pass the result of execution to the client
- ▶ Check error state.

Requirement – Socket (4/4)

▶ Client side

- ▶ Connect to server.
- ▶ Convert user command to FTP command.
- ▶ When User command is quit, convert QUIT, and quit.
- ▶ When User push ctrl+c(SIGINT), convert QUIT, and quit.
- ▶ Check error state.

Requirement – Command Conversion (1/2)

▶ Command conversion

- ▶ Client side.
- ▶ command
 - ▶ ls
 - ▶ pwd
 - ▶ dir
 - ▶ cd
 - ▶ mkdir
 - ▶ delete
 - ▶ rmdir
 - ▶ rename
 - ▶ quit
- ▶ user command will show up in server with executed child pid
 - ▶ Ex)

cli[2883]

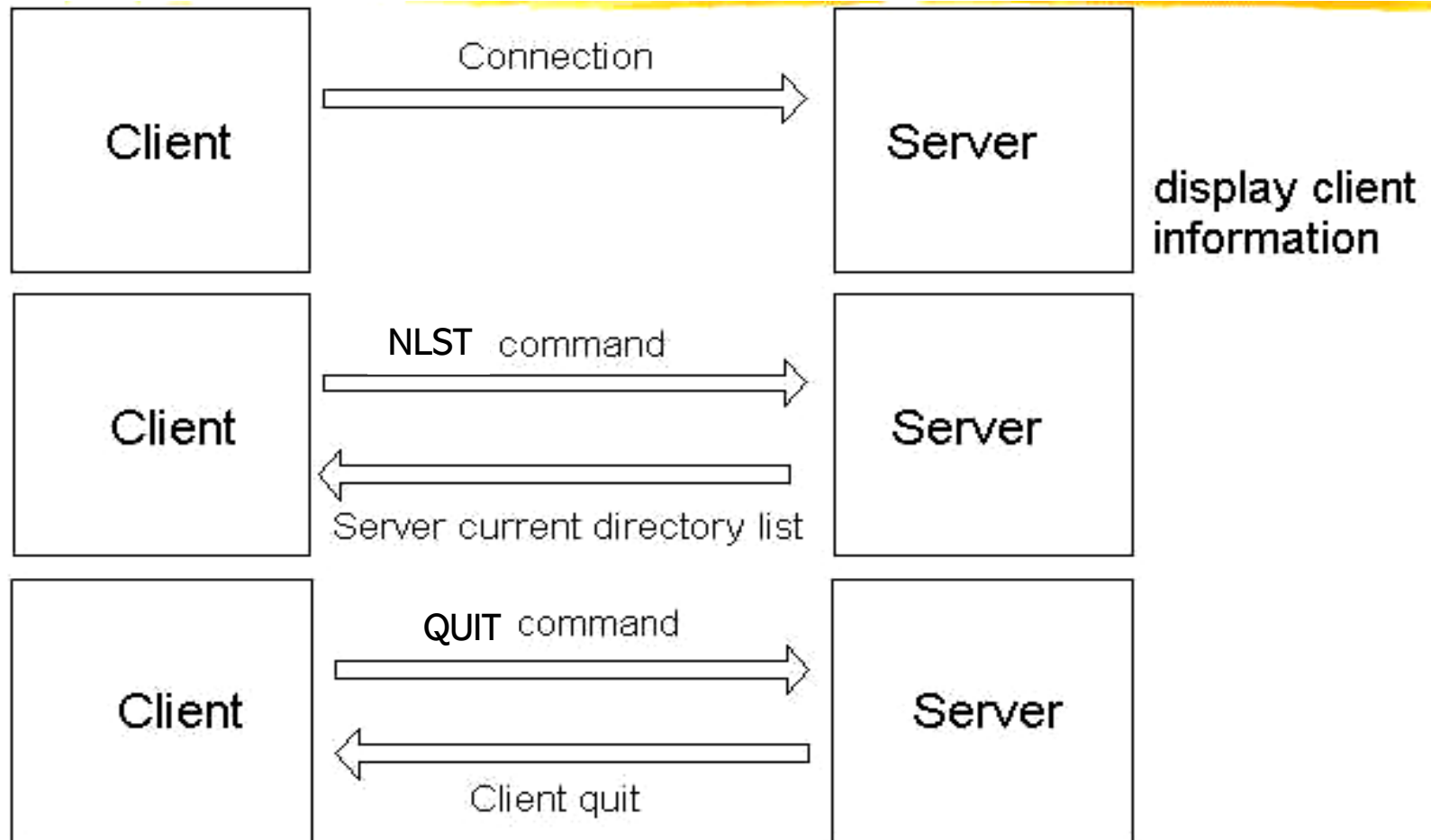
> ls

cli	cli.c	srv	srv.c	xxxx.pdf
-----	-------	-----	-------	----------

srv

> NLST [2883]

Requirement – Command Conversion (2/2)



Requirement – Concurrent Server (1/3)

▶ Concurrent Server – using fork()

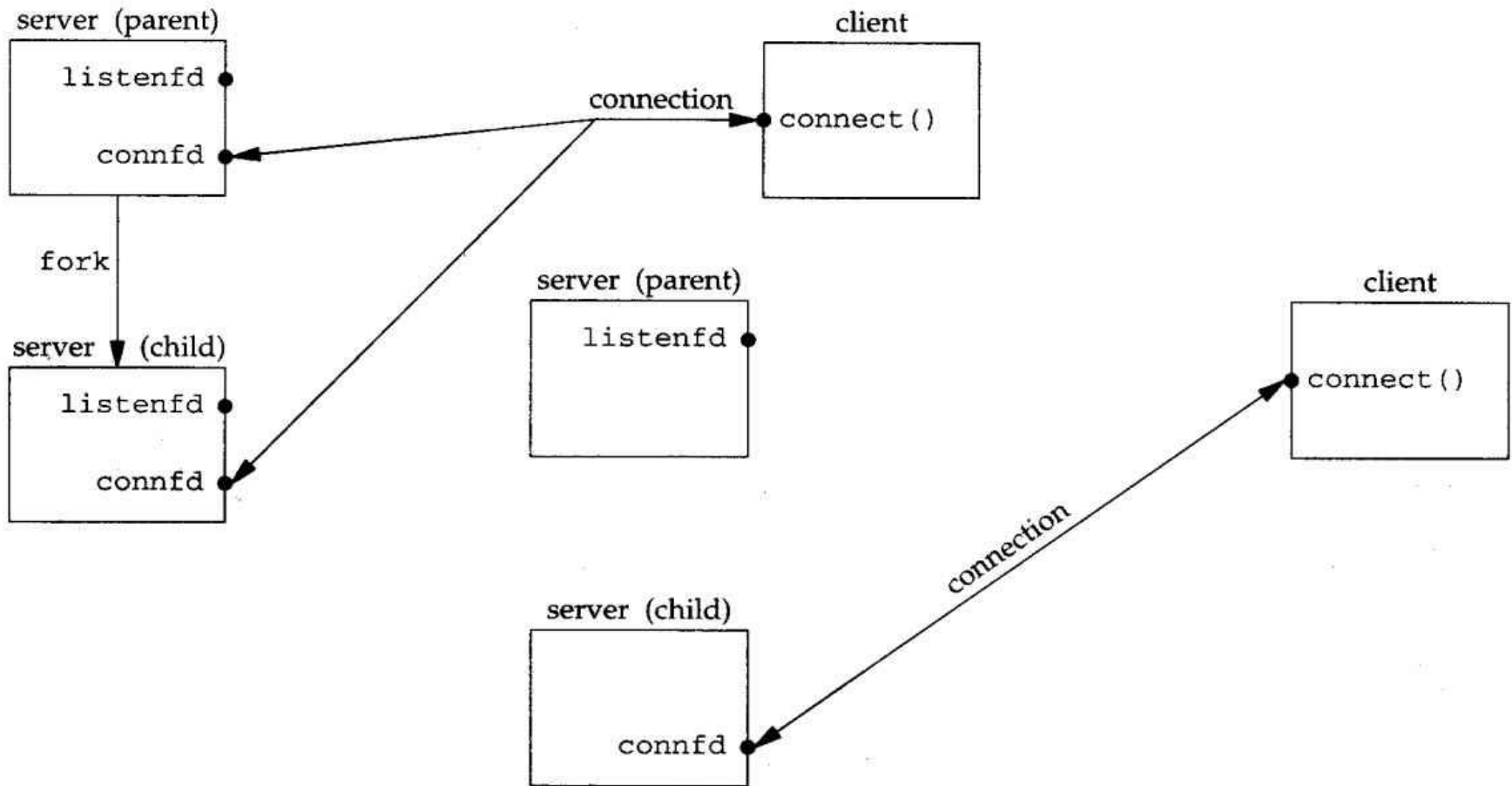
```
#include <sys/types.h>
```

```
#include <unistd.h>
```

```
pid_t fork(void);
```

- ▶ **The only way in Unix to create a new process, called once but returns twice.**
- ▶ **Returns**
 - ▶ The return value in the child is 0
 - ▶ The return value in the parent is the process ID of the new child
 - ▶ The return value on error is -1

Requirement – Concurrent Server (2/3)



Requirement – Concurrent Server (3/3)

```
pid_t pid;
int listenfd, connfd;
listenfd = socket(,,,);
bind(listenfd, ..);
listen(listenfd, LISTENQ);
for( ; ; ) {
    connfd = Accept(listenfd, ,,, );
    if( ( pid = fork() ) == 0) {
        close(listenfd);
        # 반복해서 client에서 socket를 전달받으며 해당하는
        command를 execute
        close(connfd);
        exit(0);
    }
    close(connfd);
}
```

Requirement – Signal Process

- ▶ Signal process

```
#include <signal.h>
```

```
void (*signal(int signo, void(*func) (int) ));
```

```
Returns : SIG_ERR if error
```

- ▶ Use SIGCHLD, SIGALRM, SIGINT

Requirement – Etc

▶ 다중접속 허용 & 시그널 처리

- ▶ 접속 직후, child process 의 PID 출력
- ▶ 10초 간격으로 현재 child process 의 개수와 process들의 정보를 출력
 - ▶ Process 정보 – PID, Port번호(client의 Port number), 서비스 타임(접속직후 카운트)
 - ▶ 단, 새로운 client 가 접속할 경우, 출력과 동시에 10초 interval은 그 순간부터 다시 카운트

▶ Client 프로그램 종료 시

- ▶ Client side
 - QUIT 명령어 또는 ctrl+c를 통해 종료 후, 서버에게 QUIT를 통해 알림
- ▶ Server side
 - QUIT 메시지를 받을 경우 해당 client와 연결된 child process 종료
 - 해당 process PID를 출력
 - 해당 Process의 정보는 10초마다 출력되는 process들의 정보에서 반드시 제외
 - 해당 Process에 대한 종료 format 출력

▶ Server 프로그램 종료

- ▶ Ctrl + c 로 종료.
- ▶ Server 프로그램 종료 시, 아래 동작을 수행하는 SIGINT Handler가 동작해야 함.
 - 모든 client의 연결 종료
 - 모든 child process 종료

Requirement – Sample Result

cli#1

1. cli#1 접속
6. "quit" 로 종료

cli#2

2. cli#2 접속
5. ctrl+c 로 종료

cli#3

3. cli#3 접속 후, 명령어 대기

The screenshot displays three terminal windows. The leftmost window shows the CLI interface where three clients (cli#1, cli#2, cli#3) are added and then released. The rightmost window shows the server's response, which includes client information (IP, port) and a table of child process IDs, ports, and times. The server's response is divided into sections for each client, showing the current number of clients and the release of each client.

Client	Client IP	Client Port	Child Process ID	Current Number of Client	PID	PORT	TIME
cli#1	127.0.0.1	32824	2876	1	2876	32824	1
cli#2	127.0.0.1	32826	2878	2	2876	32824	3
cli#3	127.0.0.1	32827	2880	3	2876	32824	5
cli#3	127.0.0.1	32827	2880	3	2878	32826	3
cli#3	127.0.0.1	32827	2880	3	2880	32827	1
cli#3	127.0.0.1	32827	2880	3	2876	32824	15
cli#3	127.0.0.1	32827	2880	3	2878	32826	13
cli#3	127.0.0.1	32827	2880	3	2880	32827	11
cli#3	127.0.0.1	32827	2880	3	2876	32824	25
cli#3	127.0.0.1	32827	2880	3	2878	32826	21
cli#3	127.0.0.1	32827	2880	3	2880	32827	31