KWANGWOON UNIVERSITY

2019년 1학기 **시스템프로그래밍실습** 11주차

Advanced web server

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

Assignment#3 – description

- **HTTP/HTML Basic (Assignment 3-1)**
  - HTML-ls
- **Socket System Call (Assignment 3-2)**
  - HTTP Protocol
  - socket(), bind(), listen(), accept(), connect()
- **Access Control (Assignment 3-3)**
  - String pattern matching

Process

- **A program in execution**

- **Process identifiers**
  - Every process has a unique process ID (a nonnegative integer.)

- **Create a child process using fork()**
  - Parent process
    - fork()를 호출
  - Child process
    - fork()에 의해 생성된 process

fork() System Call

```
#include  <unistd.h>

pid_t fork(void);
```

- The only way in Unix to create a new process
  - It creates a new process by duplicating the calling process.
    - The child gets a copy of the parent's data and stack.
    - The parent and child can each execute different sections of code at the same time
  - The child and parent continue to execute with the instruction that follows the call to **fork()**.

- Called once, but returns twice
  - In the **child** process, 0 is returned.
  - In the **parent** process, process ID of the new child is returned.

  - On error, -1 is returned.

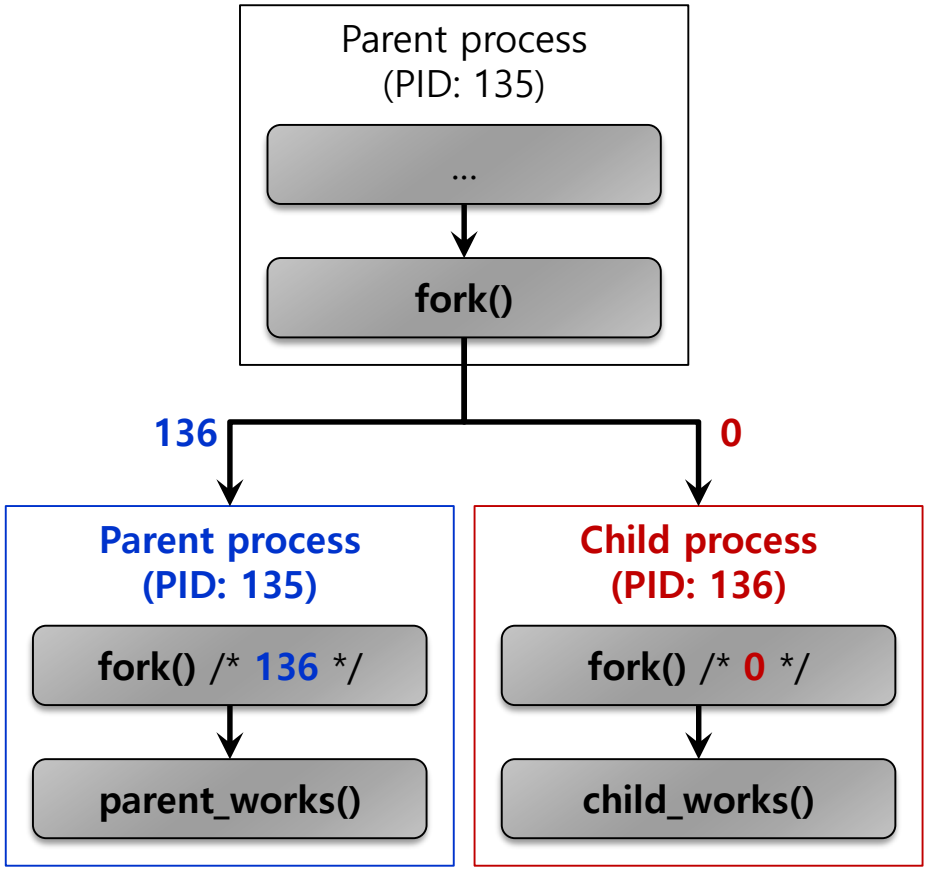
fork() System Call (cont'd)

- **Example**

```
pid_t PID = fork();

if(PID == 0)
{
    child_works();
    /* Execution codes for child process */
}

else if (PID > 0)
{
    parent_works();
    /* Execution codes for parent process */
}
```



Signal Handling

- **Signals**
  - Software interrupt
  - It provides a way of handling asynchronous events.
    - Example
      - Users press certain terminal key.
      - Hardware exceptions
      - kill() function is called.
      - Software conditions, …
  - When a signal occurs, process can
    - ignore the signal,
    - catch the signal,
    - or let the default action apply.

Signal Handling (cont'd)

- **SIGCHLD**
  - It occurs to parent process when status of child process(es) is changed.
  - Default action: ignorance
  - Generally one of the wait() functions is called
    - to fetch the child's process ID and termination status.

- **SIGALRM**
  - Time out
  - Default action: termination
  - alarm()

```
#include <signal.h>

unsigned int alarm(unsigned int seconds);
```

    - Generally it is used to generate SIGALRM.
    - In any event any previously set alarm() is canceled.

Signal Handling (cont'd)

- **Apply a new signal handler for a signal**

```
#include <signal.h>

typedef void (*sighandler_t)(int);
sighandler_t signal(int signum, sighandler_t handler);
```

  - Return values
    - If OK, Previous disposition of signal
    - On error, SIG_ERR

Process Termination

- **SIGCHLD signal**
  - It occurs to parent process when status of child process is changed.
    - The child terminated.
    - The child was stopped by a signal.
    - The child was resumed by a signal.
  - Default action : ignorance

- **Zombie process**
  - A child that terminates but has not been waited for becomes a "zombie."
  - 즉, 부모 프로세스가 자식 프로세스의 종료를 기다리지 않고 종료되어 자원을 반납하지 못하는 **자식 프로세스**

Process Termination

- **wait(), waitpid()**

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

pid_t wait(int *status);
pid_t waitpid(pid_t pid, int *status, int options);
```

- They are used to wait for state changes in a child of the calling process.
- They obtain information about the child whose state has changed.

- Behaviors
  - 자식 프로세스의 상태 변화가 없는 경우
    ➔ block
  - 호출 시점에 이미 자식 프로세스가 종료된 경우
    ➔ 종료 상태 반환
    ➔ 자식 프로세스가 사용한 자원을 모두 해제

Process Termination

- **wait(), waitpid()**

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

pid_t wait(int *status);
pid_t waitpid(pid_t pid, int *status, int options);
```

  - int options
    - WNOHANG : return immediately if no child has exited.

  - Return values
    - wait()
      - If OK, The process ID of the child whose state has changed.
      - On error, -1
    - waitpid()
      - If OK, The process ID of the child whose state has changed.
      - If options is WNOHANG and no child specified by id has yet changed state, 0
      - On error, -1

Signal Handling

Signal	Standard	Action	Comment
SIGABRT	P1990	Core	Abort signal from abort(3)
SIGALRM	P1990	Term	Timer signal from alarm(2)
SIGBUS	P2001	Core	Bus error (bad memory access)
SIGCHLD	P1990	Ign	Child stopped or terminated
SIGCLD	-	Ign	A synonym for SIGCHLD
SIGCONT	P1990	Cont	Continue if stopped
SIGEMT	-	Term	Emulator trap
SIGFPE	P1990	Core	Floating-point exception
SIGHUP	P1990	Term	Hangup detected on controlling terminal or death of controlling process
SIGILL	P1990	Core	Illegal Instruction
SIGINFO	-		A synonym for SIGPWR
SIGINT	P1990	Term	Interrupt from keyboard
SIGIO	-	Term	I/O now possible (4.2BSD)
SIGIOT	-	Core	IOT trap. A synonym for SIGABRT
SIGKILL	P1990	Term	Kill signal
SIGLOST	-	Term	File lock lost (unused)
SIGPIPE	P1990	Term	Broken pipe: write to pipe with no readers; see pipe(7)
SIGPOLL	P2001	Term	Pollable event (Sys V). Synonym for SIGIO
SIGPROF	P2001	Term	Profiling timer expired
SIGPWR	-	Term	Power failure (System V)
SIGQUIT	P1990	Core	Quit from keyboard
SIGSEGV	P1990	Core	Invalid memory reference
SIGSTKFLT	-	Term	Stack fault on coprocessor (unused)
SIGSTOP	P1990	Stop	Stop process
SIGTSTP	P1990	Stop	Stop typed at terminal
SIGSYS	P2001	Core	Bad system call (SVr4); see also seccomp(2)
SIGTERM	P1990	Term	Termination signal

⋮

* reference: http://man7.org/linux/man-pages/man7/signal.7.html

Get IP Address of Client

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

char *inet_ntoa(struct in_addr in);
```

- 32-bit의 IP 주소를 Dotted-decimal 문자열 형태의 주소(e.g. 128.124.52.62)로 변환
- struct in_addr in
  - sockaddr_in 구조체의 sin_addr 이용

KWANGWOON UNIVERSITY

실습

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

signal.c

```c
#include <signal.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

void signalHandler(int sig)
{
        if (sig == SIGINT)
        {
                printf(" signal : SIGINT \n");
                exit(0);
        }
        if (sig == SIGTSTP)
        {
                printf(" signal : SIGTSTP \n");
                exit(0);

        }
        if (sig == SIGCHLD)
        {
                printf("The child process is dead. \n");

        }
}

int main()
{
        signal(SIGCHLD, signalHandler);
        printf("input Ctrl+C or Ctrl+Z \n\n");

        int status;
        pid_t pid;
        if ((pid = fork()) == 0) //child process
        {
                signal(SIGINT, signalHandler);
                signal(SIGTSTP, signalHandler);
                while (1);

        }
        else //parent process
        {
                signal(SIGINT, SIG_IGN);
                signal(SIGTSTP, SIG_IGN);
                wait(&status);
                printf("After returning the resource, the parent process also dies. \n");

        }
}
```

KWANGWOON UNIVERSITY

Result

```
azx1593@ubuntu:~/test$ ls signal
signal
azx1593@ubuntu:~/test$ ./signal
input Ctrl+C or Ctrl+Z

^C signal : SIGINT
The child process is dead.
After returning the resource, the parent process also dies.
azx1593@ubuntu:~/test$ ./signal
input Ctrl+C or Ctrl+Z

^Z signal : SIGTSTP
The child process is dead.
After returning the resource, the parent process also dies.
azx1593@ubuntu:~/test$
```

- ctrl + c
  - SIGINT signal을 발생시킴
  - 자식 프로세스 죽을 때까지 부모 프로세스가 wait 후 종료
- ctrl + z
  - SIGTSTP signal을 발생시킴
  - 프로세스 중지

Advanced Server – “adv_server.c”

```c
#include <stdio.h>
#include <string.h>     // bzero(), ...
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>     // STDOUT_FILENO, ...

#define BUFFSIZE        1024
#define PORTNO          40000

int main()
{
        struct sockaddr_in server_addr, client_addr;
        int socket_fd, client_fd;
        int len, len_out;
        char buf[BUFFSIZE];

        if ((socket_fd = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
                printf("Server: Can't open stream socket.");
                return 0;
        }

        bzero((char *)&server_addr, sizeof(server_addr));
        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        server_addr.sin_port = htons(PORTNO);
        if (bind(socket_fd, (struct sockaddr *)&server_addr,
                                sizeof(server_addr)) < 0) {
                printf("Server: Can't bind local address.\n");
                return 0;
        }
```

* When **INADDR_ANY** is specified in the bind call, the socket will be bound to all local interfaces.

Advanced Server – “adv_server.c” (cont’d)

```
32
33          listen(socket_fd, 5);
34          while(1) {
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54          }
55          close(socket_fd);
56          return 0;
57  }
58
```

/** Fill Your Codes **/

KWANGWOON UNIVERSITY

Assignment 3-3

System Software Laboratory
College of Software and Convergence
Kwangwoon Univ.

Assignment 3-3

- **다중 접속과 접근 제어를 지원하는 웹 서버 프로그램 작성**
  - 이전 과제의 웹 서버(Assignment 3-2)의 요구조건을 만족해야 함

  - 다중 접속 지원
    - 클라이언트와 소켓 연결된 이후의 작업을 새로운 프로세스에서 수행하여 동시에 다수의 클라이언트가 접속할 수 있도록 지원
    - 클라이언트와 접속 연결 및 해제 시 마다 클라이언트들의 정보를 아래와 같이 출력 (①)
    - 10초에 한 번씩 연결되었던 클라이언트의 정보를 일괄 출력 (②)
      - 총 연결 횟수 (“Number of request(s)”)는 서버 프로그램 실행 이후의 누적 값 유지
      - 하단의 접속 기록은 최대 10개의 최근 기록만 출력(최신 시간 순서대로 정렬)

```
============= New Client ==============
IP : 127.0.0.1
Port : 18123
=======================================

========= Disconeected Client =========
IP : 127.0.0.1
Port : 18123
=======================================

============= New Client ==============
IP : 127.0.0.1
Port : 18127
=======================================

========= Disconeected Client =========
IP : 127.0.0.1
Port : 18127
=======================================

========= Connection History ==========
Number of request(s) : 2
NO.     IP              PID     PORT    TIME
1       127.0.0.1       2876    18123   Sat May 11 17:59:08 2019
2       127.0.0.1       2878    18127   Sat May 11 17:59:08 2019
```

① (New Client ~ Disconeected Client 블록 전체)

* **NO.** : 해당 클라이언트가 몇 번째로 서버에 연결된 것인지
* **IP** : 클라이언트의 IP 주소
* **Port** : 해당 클라이언트를 위한 서버의 Port 번호
* **PID** : 해당 클라이언트를 위한 프로세스의 ID
* **Time** : 서버-클라이언트가 연결된 시간

Assignment 3-3 (cont'd)

- **접근 제어**
  - 미리 허용한 IP를 가진 사용자만 서버에 접속할 수 있도록 허가
    - 현재 연결을 요청한 클라이언트의 IP를 기준으로 판단
  - 접근 가능한 클라이언트의 목록을 파일("accessible.usr")로 유지
    - Dotted-decimal 형태의 IP(ex. 128.134.52.62)를 명시
    - 10진수 수 대신에 *가 들어가 있을 경우도 처리
  - 예시

```
kks@sslab-splab-ubuntu:~$ cat accessible.usr
128.134.52.61
192.168.*.1
128.*.*.62
kks@sslab-splab-ubuntu:~$
```

Assignment 3-3 (cont'd)

- **허가되지 않은 IP에서 접속할 경우, 에러 메시지가 포함된 페이지 출력**

```
kks@sslab-splab-ubuntu:~$ cat accessible.usr
128.134.52.61
kks@sslab-splab-ubuntu:~$
```

- 허가 되지 않은 IP(e.g. 128.134.52.62)가 웹 서버(e.g. 128.134.52.61:44444)에 접근하고자 할 때 접근 제한 메시지를 출력
- 해당하는 문자열을 전부 출력해야 함
  - 단, 디자인(글자 크기 등)은 채점 요소가 아님

Assignment 3-3 (cont'd)

- **Code Requirements**
  - 이전 과제 부분에 문제가 있는 경우 감점
  - 출력 형식에 맞지 않으면 감점

- **Makefile Requirements**
  - 실행 파일이 "요일_3-3_학번"로 생성되도록 Makefile 작성
  - "$ make" 를 통해 실행파일이 생성되지 않는 경우, 0점

Assignment 3-3

- **Softcopy Upload**
  - 제출 파일
    - 보고서 (파일명 : 요일_3-3_학번.pdf)
    - 요일_3-3_학번.c , Makefile (실행 파일명 : 요일_3-3_학번)
  - Soruce code copy 적발 시 예외 없이 0점 처리
  - 위 파일들을 압축해서 제출 (파일명: 실습 요일_3-3_학번.tar.gz)
    - e.g. 월1,2 ➔ **mon**_3-3_2017202000.**tar.gz**
    - E.g. 화3,4 ➔ **tue**_3-3_2017202000.**tar.gz**
    - E.g. 금5,6 ➔ **fri**_3-3_2017202000.**tar.gz**
  - U-Campus의 과제 제출에 5월 24일(금) 23:59:59까지 제출
    - U-Campus에 올린 후 다시 다운로드 받아서 파일이 정확한지 확인
  - 미리 공지한 바와 같이, delay 받지 않음 (예외 없음)
  - Ubuntu 16.04 64bits 환경에서 채점

- **과제 질문 관련**
  - 해당 과제 출제 담당 조교에게 이메일로 문의 ➔ **조수익 조교 (azx1593@kw.ac.kr)**
  - 과제 제출 마감 당일에는 오후 4시까지 도착한 질문 메일에만 답변

Assignment 3-3 (cont'd)

- **표지**
  - 다음의 내용은 필히 기록
    - 과제 이름 (e.g. Assignment#3-3)
    - 분반 (요일, 담당 교수님)
    - 본인 인적 사항 (학번, 이름)

- **과제 내용**
  - Introduction : 5줄 이하
  - Flow chart : 4주차 자료의 Appendix 참고
  - Pseudo code : 4주차 자료의 Appendix 참고
  - Result : 수행한 내용을 캡쳐 이미지와 함께 설명
  - Conclusion : 결론 및 고찰

- **보고서 이름은 "실습 요일_과제명_학번"으로 수정**
  - e.g. 월1,2 ➔ **mon**_3-3_2019202000.**pdf**
  - e.g. 화3,4 ➔ **tue**_3-3_2019202000.**pdf**
  - e.g. 금5,6 ➔ **fri**_3-3_2019202000.**pdf**

3차 퀴즈

- **시간 및 장소**
  - 시간 : 5월 25일 토요일 오전 11시 ~ 12시
    - 오전 11시 30분 이후 입실 불가
  - 장소 : 새빛관 102호, 새빛관 205호
    - 자세한 내용은 공지 확인

- **내용**
  - 과제
    - 3차 과제
  - 강의자료
    - 8주차
    - 9주차
    - 10주차