Socket Programming

Web Crawler

wget

- wget
  - Usage:

```
WGET(1)                                GNU Wget                                WGET(1)

NAME
       Wget - The non-interactive network downloader.

SYNOPSIS
       wget [option]... [URL]...

DESCRIPTION
       GNU Wget is a free utility for non-interactive download of files from
       the Web.  It supports HTTP, HTTPS, and FTP protocols, as well as
       retrieval through HTTP proxies.

       Wget is non-interactive, meaning that it can work in the background,
       while the user is not logged on.  This allows you to start a retrieval
       and disconnect from the system, letting Wget finish the work.  By
       contrast, most of the Web browsers require constant user's presence,
       which can be a great hindrance when transferring a lot of data.

       Wget can follow links in HTML, XHTML, and CSS pages, to create local
       versions of remote web sites, fully recreating the directory structure
       of the original site.  This is sometimes referred to as "recursive
       downloading."  While doing that, Wget respects the Robot Exclusion
 Manual page wget(1) line 1 (press h for help or q to quit)
```

wget

- wget
  - Usage:

```
   Download Options
       --bind-address=ADDRESS
           When making client TCP/IP connections, bind to ADDRESS on the local
           machine.  ADDRESS may be specified as a hostname or IP address.
           This option can be useful if your machine is bound to multiple IPs.

       -t number
       --tries=number
           Set number of tries to number. Specify 0 or inf for infinite
           retrying.  The default is to retry 20 times, with the exception of
           fatal errors like "connection refused" or "not found" (404), which
           are not retried.

       -O file
       --output-document=file
           The documents will not be written to the appropriate files, but all
           will be concatenated together and written to file.  If - is used as
           file, documents will be printed to standard output, disabling link
           conversion.  (Use ./- to print to a file literally named -.)

           Use of -O is not intended to mean simply "use the name file instead
           of the one in the URL;" rather, it is analogous to shell
           redirection: wget -O file http://foo is intended to work like wget
Manual page wget(1) line 200 (press h for help or q to quit)
```

wget

- wget
  - Example:
    - $ wget www.naver.com -O naver.html

```
pi@raspberrypi:~/Socket $ wget www.naver.com -O naver.html
--2017-09-24 00:49:10--  http://www.naver.com/
Resolving www.naver.com (www.naver.com)... 202.179.177.22, 202.179.177.21
Connecting to www.naver.com (www.naver.com)|202.179.177.22|:80... connected.
HTTP request sent, awaiting response... 302 Moved Temporarily
Location: https://www.naver.com/ [following]
--2017-09-24 00:49:10--  https://www.naver.com/
Connecting to www.naver.com (www.naver.com)|202.179.177.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘naver.html’

naver.html              [ <=>                ] 152.51K  --.-KB/s   in 0.1s

2017-09-24 00:49:10 (1.07 MB/s) - ‘naver.html’ saved [156171]

pi@raspberrypi:~/Socket $ ls -lh naver.html
-rw-r--r-- 1 pi pi 153K Sep 24 00:49 naver.html
pi@raspberrypi:~/Socket $
```

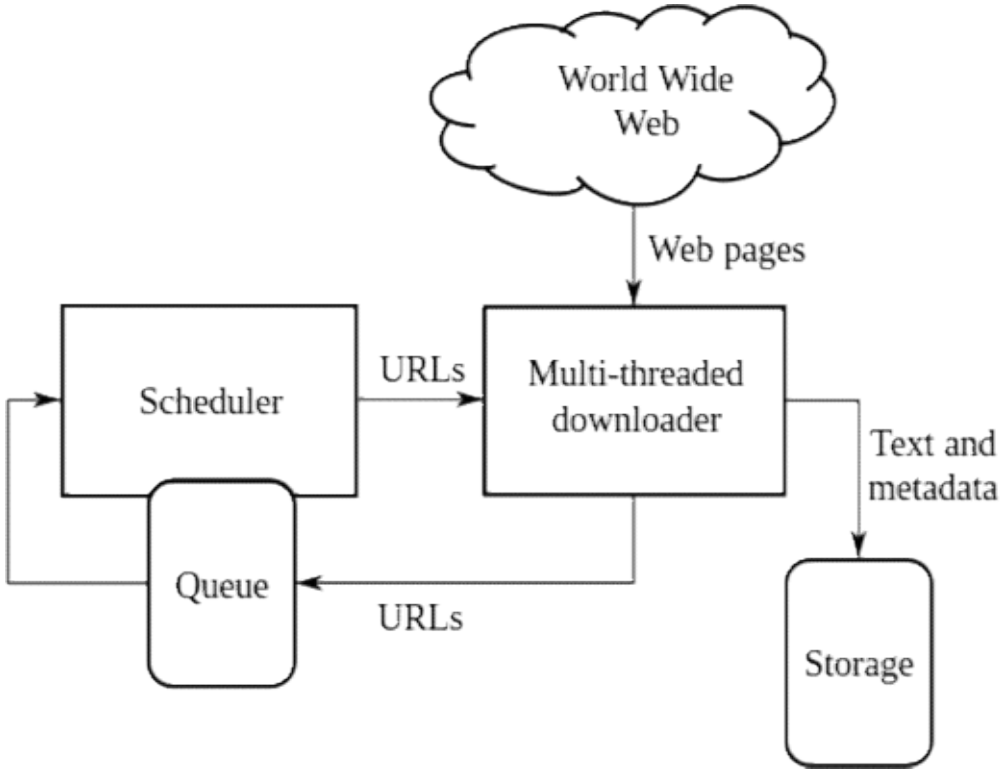
wget

- wget
  - Contents of a downloaded file (naver.com)

```
<!doctype html>
<html lang="ko" class="svgless">
<head>
<meta charset="utf-8">
<meta name="Referrer" content="origin">
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=1100">
<meta name="apple-mobile-web-app-title" content="NAVER" />
<meta property="og:title" content="네이버">
<meta property="og:url" content="http://www.naver.com/">
<meta property="og:image"
    content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png
    ">
<meta property="og:description" content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세
    요"/>
<meta name="twitter:card" content="summary">
<meta name="twitter:title" content="">
```

Web Crawler

- Web Crawler (or spidering)
  - A **Web crawler**, sometimes called a **spider**, is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing



[source: wikipedia]

Web Crawler

- How to get URLs
  - HTML 코드에 포함되는 하이퍼링크(hyperlink) 구문은
    - “http://” 또는 “https://” 로 시작됨
    - 따라서 C언어의 문자열 비교 함수인 strcmp(), strncmp() 등의 함수를 사용하여 문자열 비교가 가능함

Web Crawler

- How to get URLs
  - Example Code for getting urls:

```
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv) {
    char cmd[80];
    FILE *ifp;
    char *token;
    char delim[] = "='\"\n ";
    static char str[1024];

    if (argc != 2) {
        printf("Usage : %s\n", argv[0]);
        return -1;
    }

    sprintf(cmd, "wget %s -O index.txt", argv[1]);
    system(cmd); /* get page */

    ifp = fopen("./index.txt", "rw");
```

Web Crawler

- How to get URLs
  - Example Code for getting urls:

```
   while (fgets(str, 1024, ifp) != NULL) {
        token = strtok(str, delim);

        while ((token = strtok(NULL, delim)) != NULL) {
            if ((strncmp(token, "http:", 5) == 0) ||
                (strncmp(token, "https:", 6) == 0))
            {
                printf("token : %s\n", token);
            }
        }

        bzero(str, sizeof(str));
    }

    fclose(ifp);

    return 0;
}
```

Web Crawler

- How to get URLs
  - Results:

```
token : http://www.naver.com/
token : https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png
token : http://www.naver.com/
token : https://s.pstatic.net/static/www/mobile/edit/2016/0705/mobile_212852414260.png
token : https://s.pstatic.net/pm2/css/main_v17092101.css
token : https://s.pstatic.net/pm2/css/webfont_v170623.css
token : https://ssl.pstatic.net/sstatic/search/pc/css/api_atcmp_170914.css
token : https://s.pstatic.net/pm2/js/c/nlog_v170829.js
token : https://s.pstatic.net/pm2/js/c/jindo_170307a.js
token : http://help.naver.com/support/alias/contents2/naverhome/naverhome_1.naver
token : http://jr.naver.com
token : http://happybean.naver.com/main/SectionMain.nhn
token : https://search.naver.com/search.naver
token : https://help.naver.com/support/alias/search/word/word_16.naver
token : https://nid.naver.com/nidlogin.login
token : https://help.naver.com/support/alias/search/word/word_16.naver
token : https://help.naver.com/support/contents/contents.nhn?serviceNo
token : https://help.naver.com/support/alias/search/word/word_17.naver
token : https://help.naver.com/support/alias/search/word/word_18.naver
token : https://help.naver.com/support/alias/search/word/word_17.naver
token : https://help.naver.com/support/alias/search/word/word_18.naver
```

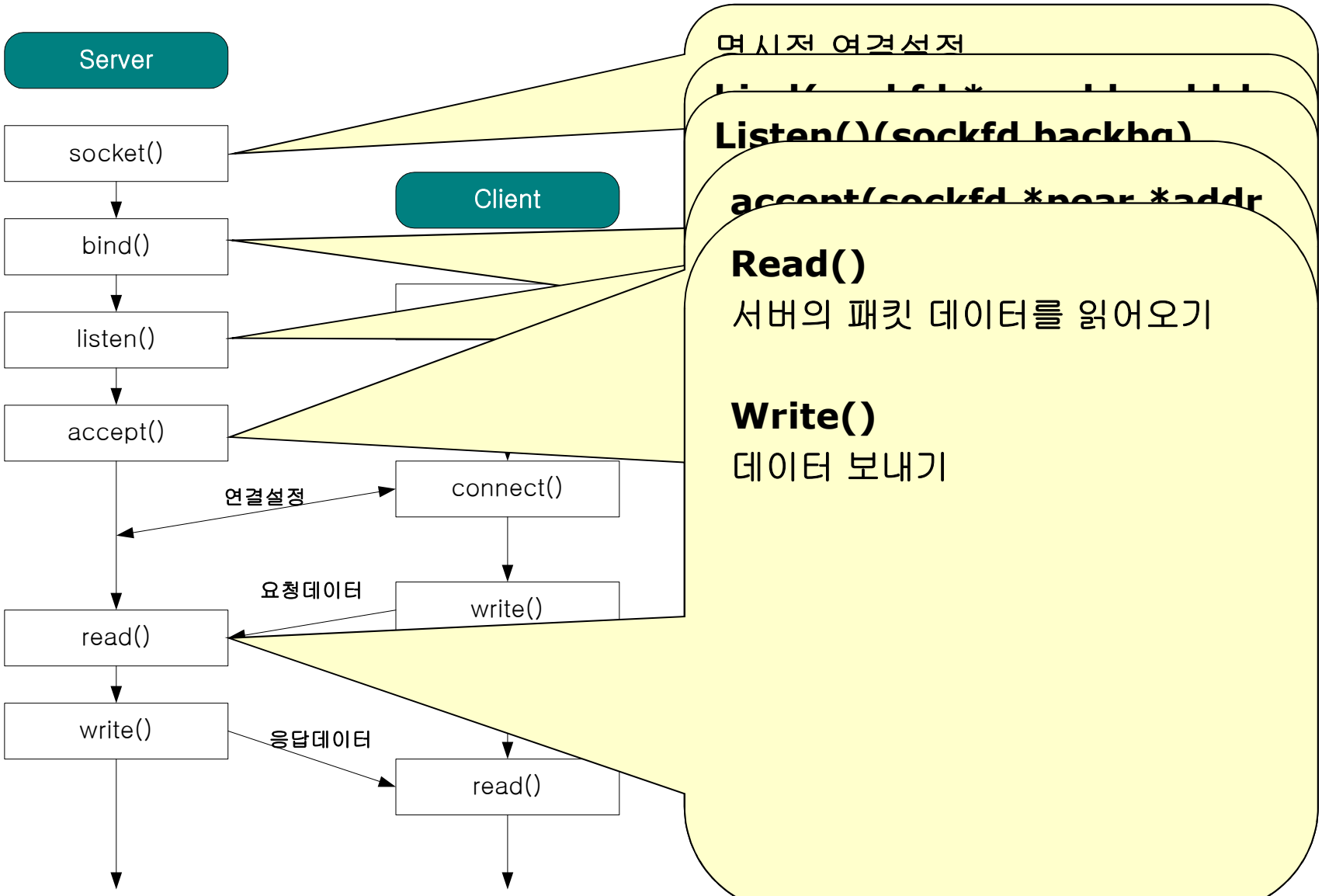
Web Crawler

- How to get links for web pages or domains

Socket Programming

- Linux/Unix BSD Socket

유닉스 프로그램 관점의 연결 시나리오



한다.

실습: simple_tcp_server

- 목 표
  - 간단한 TCP 예제를 살펴본다.
  - 소켓을 생성하는 방법을 배운다.
  - ethereal 프로그램으로 3way handshake 과정과 TCP 헤더를 살펴보자

실습: simple_tcp_server

- inet.h

```
1  /*********************************
2   * Filename: inet.h
3   * Title: Simple TCP Server
4   * Desc: Simple TCP Server
5   *********************************/
6  #include <stdio.h>
7  #include <sys/types.h>
8  #include <sys/socket.h>
9  #include <netinet/in.h>
10 #include <arpa/inet.h>
11
12 #define SERV_UDP_PORT   7000 /* UDP Server port */
13 #define SERV_TCP_PORT   7000 /* TCP Server port */
14 #define SERV_HOST_ADDR  "192.168.0.2" /* Server IP address */
15
16 char *pname;
```

실습: simple_tcp_server

- server.c

```
1  /*********************************
2   * Filename: server.c
3   * Title: Simple TCP Server
4   * Desc: Simple TCP Server
5   *********************************/
6  #include "inet.h"
7
8  int main ( int argc, char* argv[] ) {
9      int sockfd, newsockfd, clilen;
10     struct sockaddr_in  cli_addr;
11     struct sockaddr_in  serv_addr;
12     char buff[30];
13     pname = argv[0];
14
15     if ( (sockfd =                                    ) < 0 ) {
16         puts( "Server: 스트림 소켓을 열 수 없습니다.");
17         exit(1);
18     }
19
20     /* 서버의 주소를 등록하여 클라이언트가 접속 가능하게 한다. */
21     bzero((char *) &serv_addr, sizeof(serv_addr));
22     serv_addr.sin_family = AF_INET;
23     serv_addr.sin_addr.s_addr = htonl( INADDR_ANY );
24     serv_addr.sin_port = htons( SERV_TCP_PORT );
25
26     if (                                              ) < 0 ) {
27         puts( "Server: 지역어드레스로 바인딩 할 수 없습니다.");
28         exit(1);
29     }
30
```

TODO: 스트림 소켓 지정하기

TODO: 어드레스 바인딩 하기

실습: simple_tcp_server

- server.c (cont.)

```
31      TODO: 동시 접속 큐 설정하기 - 5로 설정
32      clilen = sizeof( cli_addr );
33      newsockfd = TODO: 클라이언트 요청을 받아들이기 위한 Accept ;
34
35      if ( newsockfd < 0 ) {
36          puts("Server: accept error!");
37          exit(1);
38      }
39
40      if ( read(newsockfd, buff, 20) <= 0 ) {
41          puts( "Server: readn error!");
42          exit(1);
43      }
44      printf("Server: Received String = %s \n", buff);
45      close( sockfd );
46      close( newsockfd );
47
48      return 0;
49  }
```

실습: simple_tcp_server

- client.c

```c
/*********************************
 * Filename: client.c
 * Title: Simple TCP Server
 * Desc: Simple TCP Server
 *********************************/
#include "inet.h"

int main ( int argc, char *argv[]) {
    int sockfd;
    struct sockaddr_in serv_addr;
    pname = argv[0];

    /* 연결하고자 하는 서버의 주소 및 포트번호를 구조체 serv_addr에 채운다 */
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr( SERV_HOST_ADDR );
    serv_addr.sin_port = htons( SERV_TCP_PORT );

    /* TCP Socket Open */
    if ((TODO: 스트림 소켓 열기                                    ) < 0 ) {
        puts( "Client: 소켓을 열수 없습니다.");
        exit(1);
    }

    /* 서버에 접속 */
    if ( TODO: 서버에 접속하기                                                ) < 0 ) {
        puts( "Client: 서버에 연결할 수 없습니다.");
        exit(1);
    }

```

실습: simple_tcp_server

- client.c (Cont.)

```
31 -    if ( write( sockfd, "Hello world!~", 20) < 20 ) {
32          puts( "Client: writen error" );
33          exit(1);
34      }
35
36      close( sockfd );
37  }
```

UDP

UDP Socket Program

- UDP Socket Example (Receiver 1/3)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFF_SIZE     1024
```

UDP Socket Program

- UDP Socket Example (Receiver 2/3)

```
int main(int argc, char *argv[]) {
  int server_fd, client_fd;
  int len;
  struct sockaddr_in server_addr, client_addr;
  socklen_t addrlen = sizeof(client_addr);

  unsigned char recv_buf[BUFF_SIZE];
  unsigned char send_buf[BUFF_SIZE];

  server_fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

  memset(&server_addr, 0, sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  //server_addr.sin_addr.s_addr = htonl(inet_addr("169.254.150.170"));
  server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
  server_addr.sin_port = htons(atoi(argv[1]));

  if (bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
    printf("bind error\n");
    exit(-1);
  }
```

UDP Socket Program

- UDP Socket Example (Receiver 3/3)

```
  while (1) {
    printf("Waiting data ...\n");
    fflush(stdout);

    len = recvfrom(server_fd, recv_buf, BUFF_SIZE, 0,
                   (struct sockaddr*)&client_addr, &addrlen);
    recv_buf[len] = '\0';

    printf("recv from[%d]: %s\n", client_fd, recv_buf);
    if (strncmp(recv_buf, "BYE", 3) == 0) break;

    memcpy(send_buf, recv_buf, len);
    len = sendto(server_fd, send_buf, BUFF_SIZE, 0,
                 (struct sockaddr*)&client_addr, addrlen);
  }

  close(server_fd);

  return 0;
}
```

UDP Socket Program

- UDP Socket Example (Sender 1/3)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFF_SIZE     1024
```

UDP Socket Program

- UDP Socket Example (Sender 2/3)

```
int main(int argc, char *argv[]) {
  int sock_fd;
  int len;
  struct sockaddr_in server_addr, client_addr;
  socklen_t addrlen = sizeof(server_addr);

  unsigned char recv_buf[BUFF_SIZE];
  unsigned char send_buf[BUFF_SIZE];

  sock_fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

  memset(&server_addr, 0, sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
  server_addr.sin_port = htons(atoi(argv[1]));

  if (connect(sock_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
    printf("Connection failed\n");
    exit(-1);
  }
```

UDP Socket Program

- UDP Socket Example (Sender 3/3)

```
  while (1) {
    printf("Input message send to Server.\n");
    len = read(STDIN_FILENO, send_buf, BUFF_SIZE);
    send_buf[len] = '\0';

    sendto(sock_fd, send_buf, len, 0, (struct sockaddr*)&server_addr, addrlen);

    if (strncmp(send_buf, "BYE", 3) == 0) break;

    recvfrom(sock_fd, recv_buf, BUFF_SIZE, 0,
             (struct sockaddr*)&server_addr, &addrlen);
    recv_buf[len] = '\0';

    printf("Recv from Server: %s\n", recv_buf);

    bzero(recv_buf, strlen(recv_buf));
    bzero(send_buf, strlen(send_buf));
  }

  close(sock_fd);
  return 0;
}
```

Multi-Processes

프로세스 생성

- 프로세스 생성
  - 프로그램 내에서 프로세스 생성 호출 : fork, exec..
  - 사용자가 프로세스 수행을 명령
    - shell이 프로세스 생성 호출
  - 부모 프로세스와 자식 프로세스 (트리화)
    - fork()를 이용하여 생성
  - 운영체제에서는 프로세스 테이블의 엔트리 생성
  - 수행방법
    - 부모와 자식 프로세스가 동시 수행
    - 부모는 자식 프로세스가 완료될 때 까지 대기

프로세스 생성 - fork()

- fork()
  - Fork를 호출한 프로세스를 복제하여 완전히 동일한 프로세스 생성
  - Prototype

```
#include <sys/types.h>
#include <unistd.h>
pid_t fork(void);
```

  - Return value

ret == 0 : child process

ret > 0 : parent process

ret < 0 : fork() error

Example#1: 프로세스 생성 - fork()

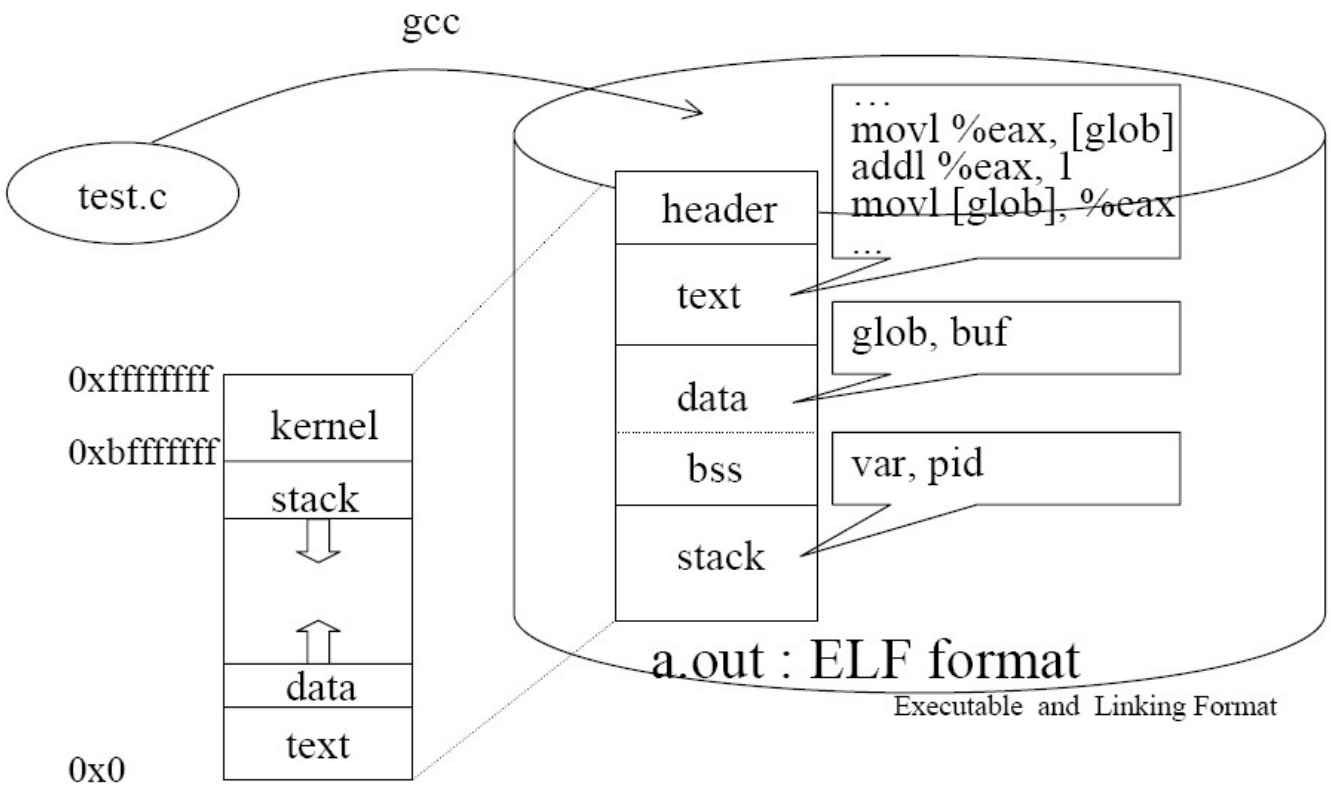
- Example

```
int glob = 6;
char buf[] = “write to stdout\n”;

int main(void)
{
    int var;
    pid_t pid;
    var = 88;

    write(STDOUT_FILENO, buf, sizeof(buf)-1);
    printf(“before fork\n”);

    if ((pid = fork()) == 0) {
        glob++, var++;
    } else sleep(2);

    printf(“pid = %d, glob = %d, var = %d\n”, getpid(), glob, var);
    exit(0);
}
```

(*source : Advanced Programming in the UNIX Environment, prog. 8.1*)

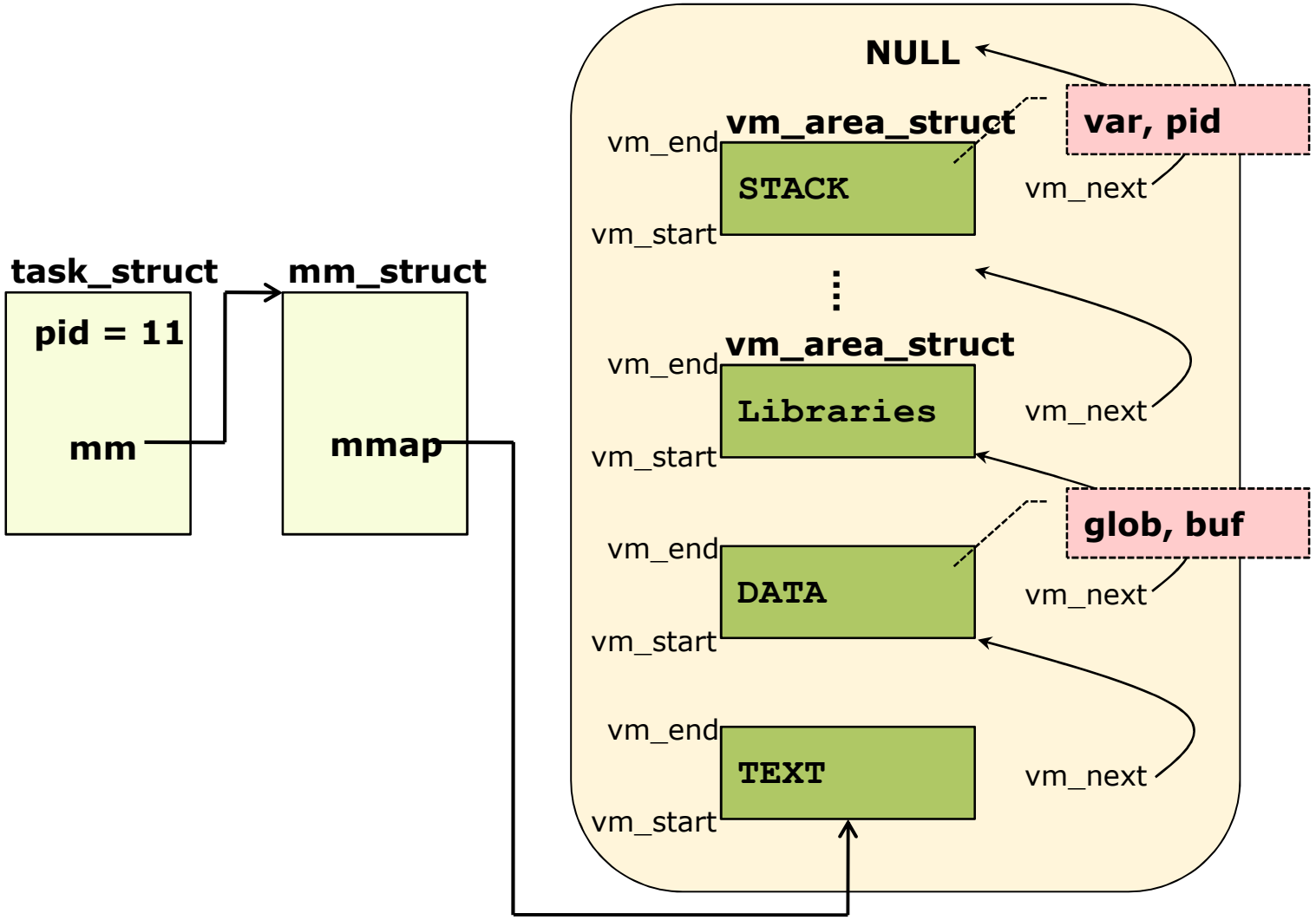
fork() Internal

- fork internal : compile results



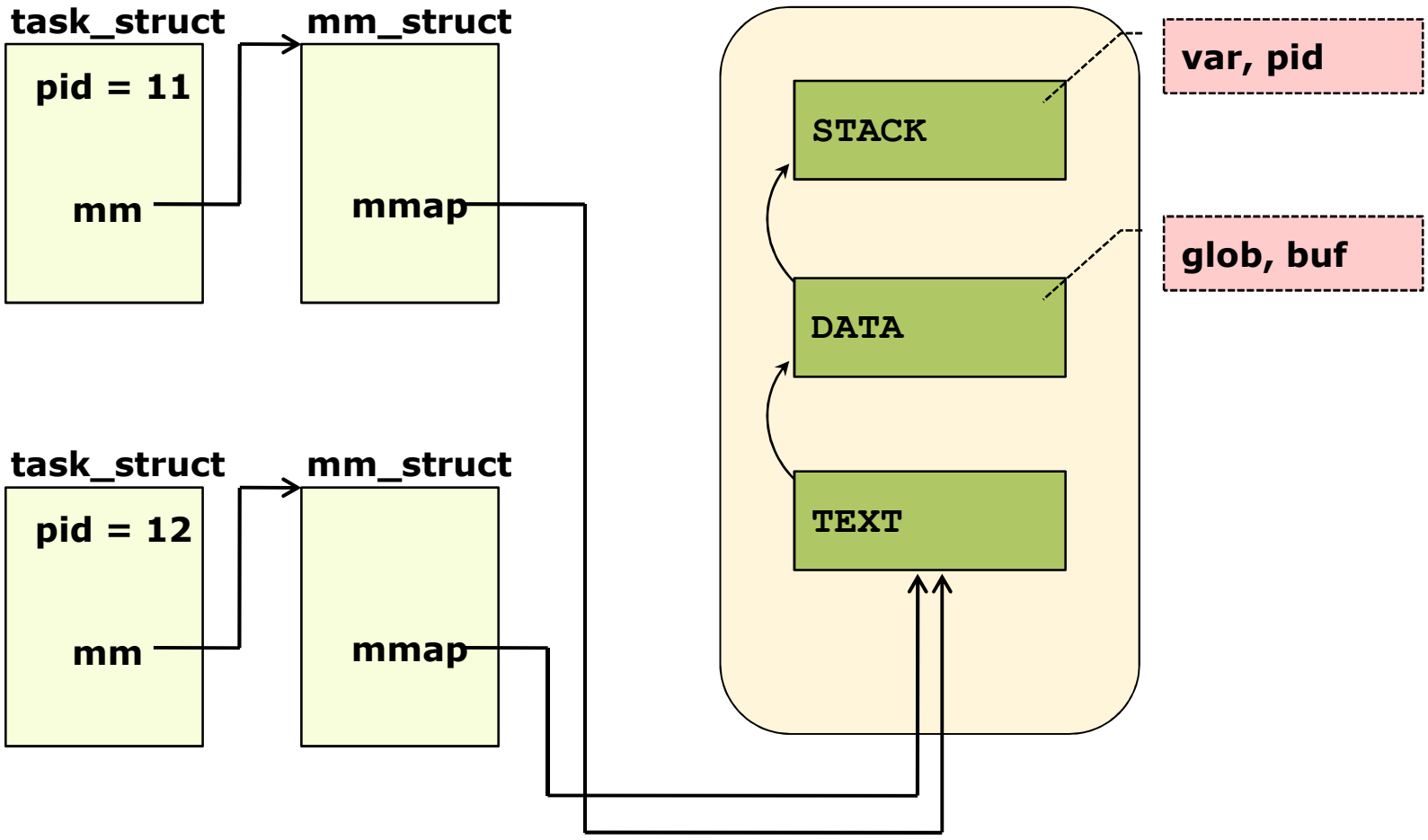
fork() Internal

- fork internal : before fork



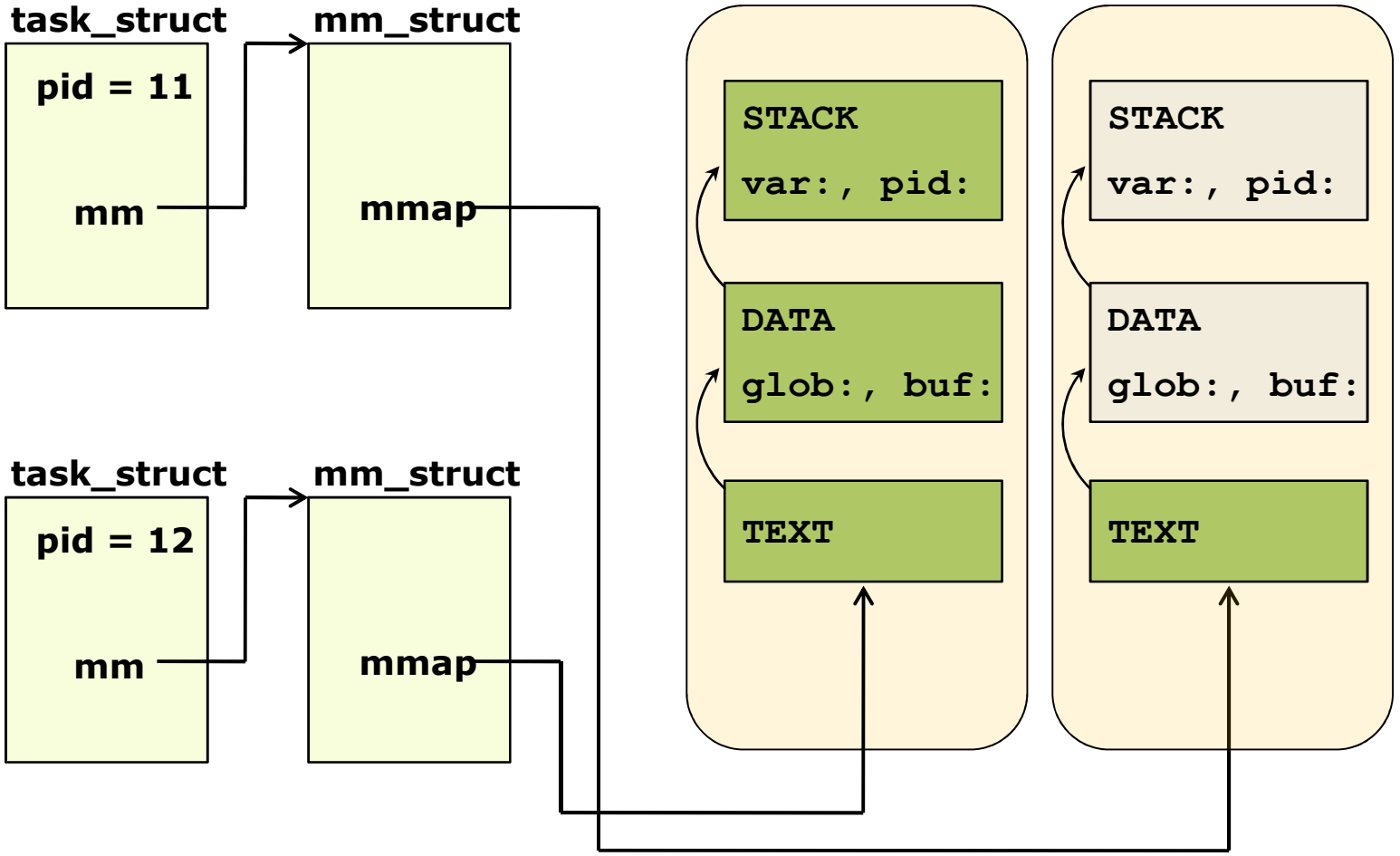
fork() Internal

- fork internal : just fork



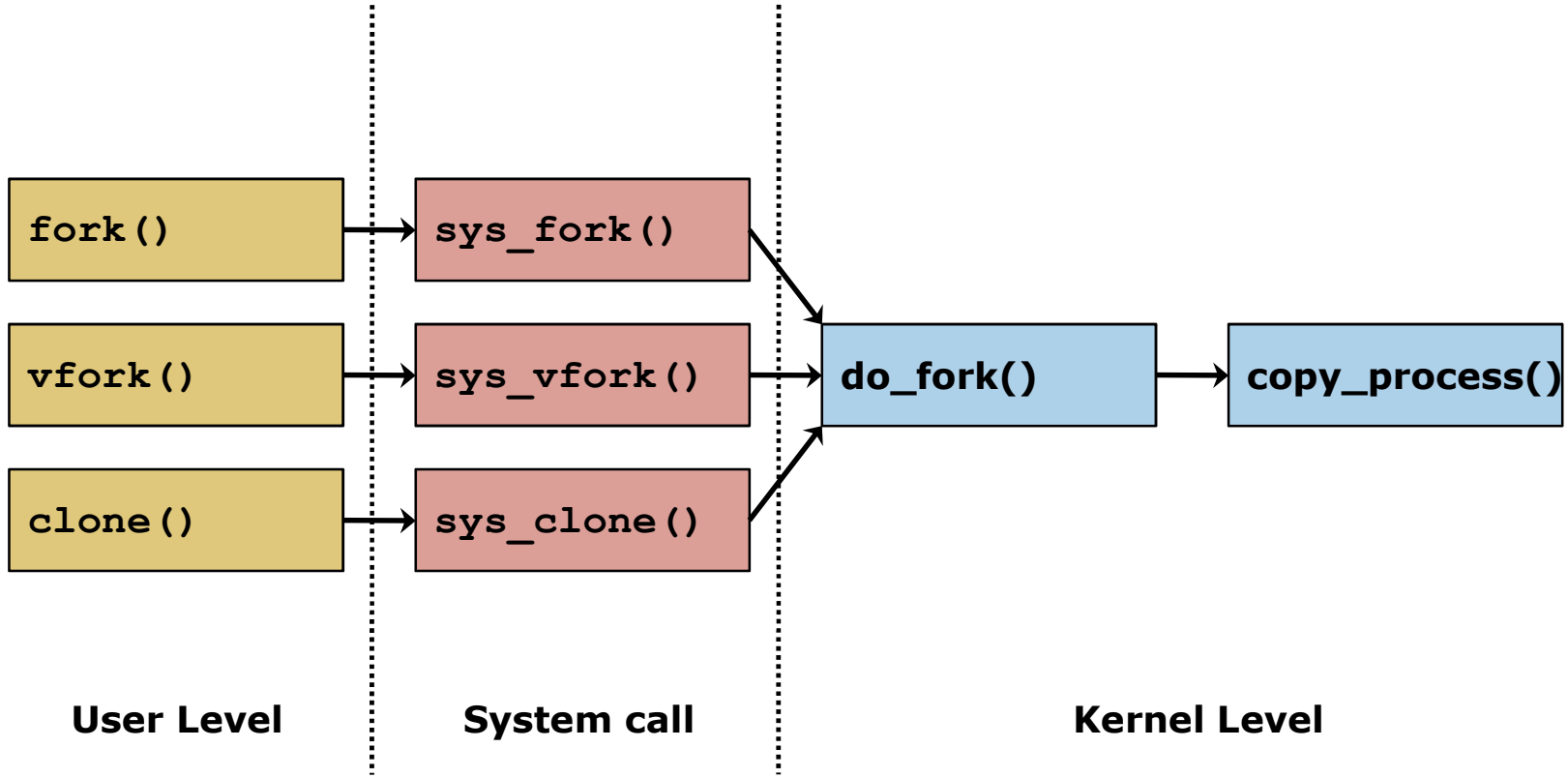
fork() Internal

- fork internal : after fork



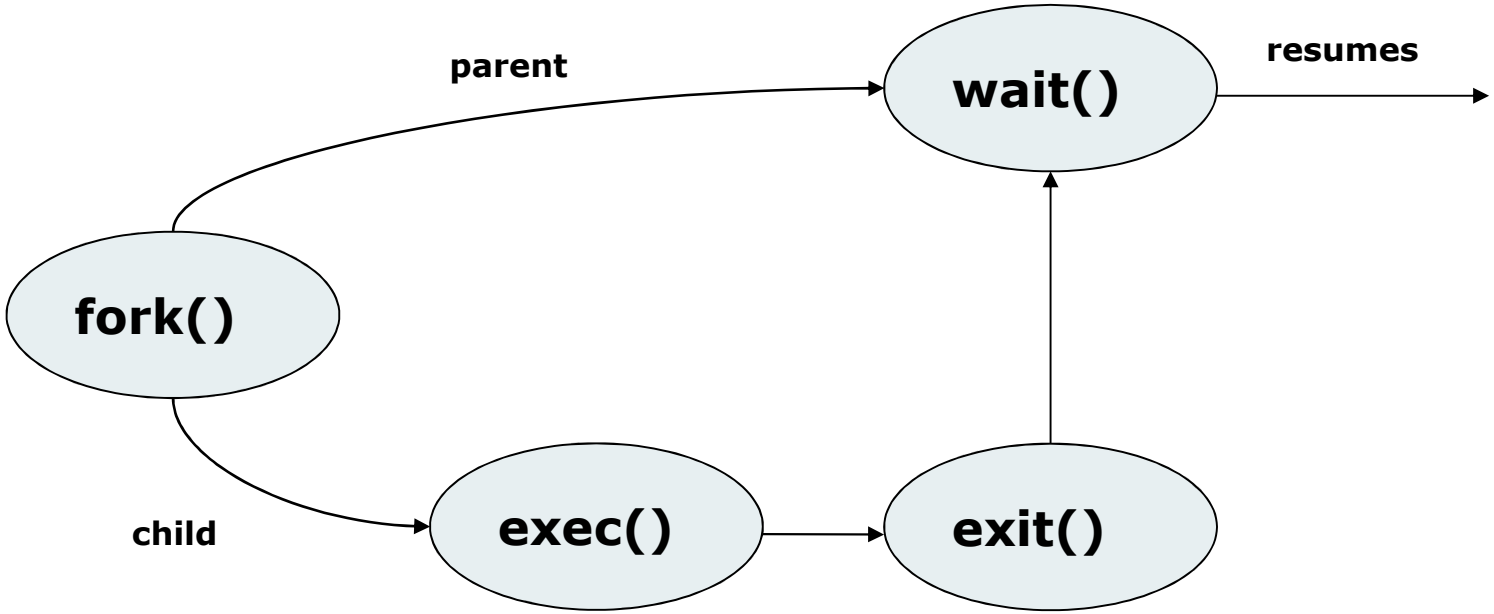
fork() Trace

- fork internal : fork() trace



프로세스 생성

- fork + exec 프로그램
  - fork : 자식 프로세스 생성
  - fork + exec : 호출하는 프로세스의 기억장소에 새 프로그램 로드



Example#2 : fork()+exec()

- fork()로 새 프로세스를 생성 예제

```
#include <stdio.h>

int main(int argc, char *argv[]) {

      int pid;
    /* fork another process */
    pid = fork();
    if(pid < 0) {                /* error occurred */
        fprintf(stderr, "Fork Failed");
        exit(-1);
    } else if (pid== 0) {   /* child process */
        execl("/bin/ls", "ls", NULL);
    } else {                          /* parent process */
        wait(NULL); //자식이 종료될 때 까지 기다린다.
        printf("Child Complete\n");
        exit(0);
    }
}
```

프로세스 종료

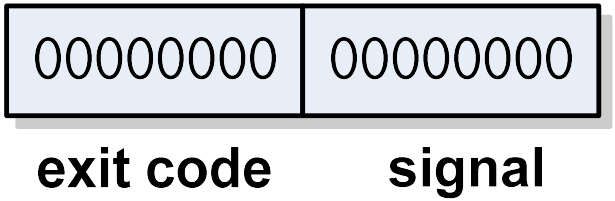
- 프로세스 종료 (Process Termination)
  - exit(): 시스템 호출
    - 계산 결과는 부모에게 리턴
    - 메모리(물리적/가상적), 오픈한 파일, I/O버퍼를 OS에게 돌려줌
    - 대부분 do_exit() 사용
    - do_exit()는 schedule()을 호출하여 새 프로세스로 분기

프로세스 종료

- 프로세스 종료 (Process Termination)
  - wait system call : return값 = 종료하는 자식의 pid
  - wait(&status) /* int status */
    - **status :**자식이 **exit**으로 종료될 때의 상태정보
      - **정상종료 경우:** 하위 8bits는 0,
        상위 8bits 는 exit status(자식 프로세스가 exit 명령으로 전달한 값),
      - **signal로 종료된 경우:** 하위 8bits는 signal 번호, 상위 8bits는 0
    - (상위 8 비트 추출)

      status >> 8;

      status &= 0xFF;



프로세스 종료

- 좀비 (zombie) 프로세스
  - 실행이 종료되어 부모 프로세스가 반환 코드를 받을 때까지 프로세스 테이블 엔트리에 정보가 남아 있는 프로세스
  - 종료된 프로세스는 오직 커널 스택과 슬랩 객체만 존재

- 고아 (orphan) 프로세스 :
  - 부모 프로세스가 먼저 종료된 경우.
  - 모든 고아 프로세스는 자동적으로 init의 "양자"가 된다
  - 2.6 커널에서는 자식 리스트와 추적중인 자식 리스트로 구분하여 관리
  - Init 프로세스는 주기적으로 wait()를 호출하여 자식 프로세스 정리

exec

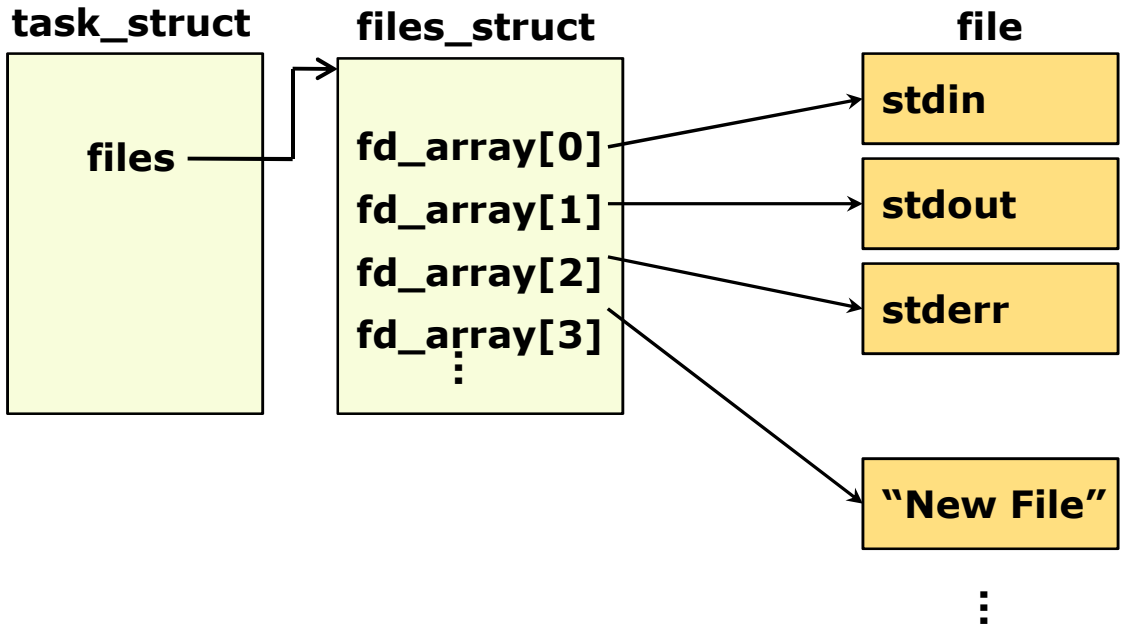
- execve()
  - 현재 프로세스 이미지를 새로운 프로세스 이미지로 교체
  - Prototype

```
#include <unistd.h>
extern char **environ;
int execl(const char *path, const char *arg, ...);
int execlp(const char *file, const char *arg, ...);
int execle(const char *path, const char *arg, ..., char *const envp[]);
int execv(const char *path, char *const argv[]);
int evecvp(const char *file, char *const argv[]);
```

  - Return value : none except error occurred

File Descriptor

- File descriptor
  - Standard input 0
  - Standard output 1
  - Standard error2



Multi-Process based Server socket

Multi-Process based Server Socket

- 다중 클라이언트 처리를 위한 서버 프로세스 분산
  - 서버 프로세스의 부하 분산 (load balancing)
  - 새로운 클라이언트가 접속할 때마다 서버 프로세스를 복제하여 각각의 클라이언트를 담당

```
while (1) {
  client_fd = accept(server_fd, (struct sockaddr*)&client_addr, &addrlen);

  if ((pid=fork()) == 0) { // child process
    printf("New client connected [%d]\n", client_fd);

    while (1) {
      len = read(client_fd, recv_buf, BUFF_SIZE);
      recv_buf[len] = '\0';

      if (strncmp(recv_buf, "BYE", 3) == 0) break;
      printf("recv from[%d]: %s\n", client_fd, recv_buf);
      len = write(client_fd, recv_buf, BUFF_SIZE);
      bzero(recv_buf, strlen(recv_buf));
    }
  }
}
```

Multi-Process based Server Socket

- 다중 클라이언트 처리를 위한 서버 프로세스 분산
  - 클라이언트가 접속하거나 종료될 때마다 현재 접속 중인 모든 클라이언트에게 접속/종료 이벤트를 전달하기 위한 방법?