

Assignment4

2016116783 김성록

Example1

fork()

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

int gval = 10;
int main(int argc, char *argv[]){
    pid_t pid;
    int lval = 20;
    gval++, lval += 5;
    pid=fork();
    if(pid==0) {
        gval+=2, lval+=2;
    }
    else {
        gval-=2, lval-=2;
    }
    if(pid==0) printf("Child Proc : [%d, %d]\n",gval,lval);
    else printf("Parent Proc : [%d, %d]\n",gval,lval);
    return 0;
}
```

```
root@eb8d501c53c5:/home# ./fork
Parent Proc : [9, 23]
Child Proc : [13, 27]
root@eb8d501c53c5:/home#
```

Example2

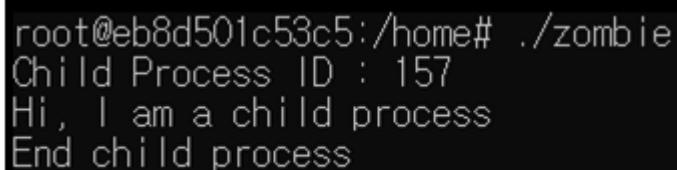
Zombie.c

```

#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[]){
    pid_t pid = fork();
    if(pid==0) {
        puts("Hi, I am a child process");
    }
    else {
        printf("Child Process ID : %d\n",pid);
        sleep(30);
    }
    if(pid==0) puts("End child process");
    else puts("End parent process");
    return 0;
}

```



```

root@eb8d501c53c5:/home# ./zombie
Child Process ID : 157
Hi, I am a child process
End child process

```

Example3

wait.c

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char *argv[]){
    int status;
    pid_t pid = fork();

    if(pid==0) {
        return 3;
    }
    else {
        printf("Child Process ID : %d\n",pid);
        pid = fork();
        if(pid==0) exit(7);
    }
}

```

```

        else{
            printf("Child Process ID : %d\n",pid);
            wait(&status);
            if(WIFEXITED(status))
                printf("Child send one : %d\n",WEXITSTATUS(status));
            wait(&status);
            if(WIFEXITED(status))
                printf("Child send two : %d\n",WEXITSTATUS(status));
            sleep(30);
        }
    }
    return 0;
}

```

```

root@eb8d501c53c5:/home# sudo ./wait
Child Process ID : 175
Child Process ID : 176
Child send one : 3
Child send two : 7

```

Example4

waitpid.c

```

#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char *argv[]){
    int status;
    pid_t pid = fork();

    if(pid==0) {
        sleep(15);
        return 24;
    }
    else {
        while(!waitpid(-1, &status, WNOHANG)){
            sleep(3);
            puts("Sleep 3 sec");
        }
        if(WIFEXITED(status))
            printf("Child send : %d\n",WEXITSTATUS(status));
    }
    return 0;
}

```

```
root@eb8d501c53c5:/home# ./waitpid
Sleep 3 sec
Sleep 3 sec
Sleep 3 sec
Sleep 3 sec
Sleep 3 sec
Child send : 24
root@eb8d501c53c5:/home#
```

Example5

signal.c

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

void timeout(int sig){
    if(sig==SIGALRM)
        puts("Time out!\n");
    alarm(2);
}

void keycontrol(int sig){
    if(sig==SIGINT)
        puts("CTRL + C pressed\n");
}

int main(int argc, char * argv[]){
    int i;
    signal(SIGALRM, timeout);
    signal(SIGINT, keycontrol);
    alarm(2);
    for (i=0; i<3; i++){
        puts("wait...\n");
        sleep(100);
    }
    return 0;
}
```

```
root@eb8d501c53c5:/home# ./signal
wait...

Time out!

wait...

Time out!

wait...

Time out!

root@eb8d501c53c5:/home# ./signal
wait...

Time out!

wait...

^CTRL + C pressed

wait...

Time out!

root@eb8d501c53c5:/home#
```

Example6

sigaction.c

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

void timeout(int sig){
    if(sig==SIGALRM)
        puts("Time out!\n");
    alarm(2);
}

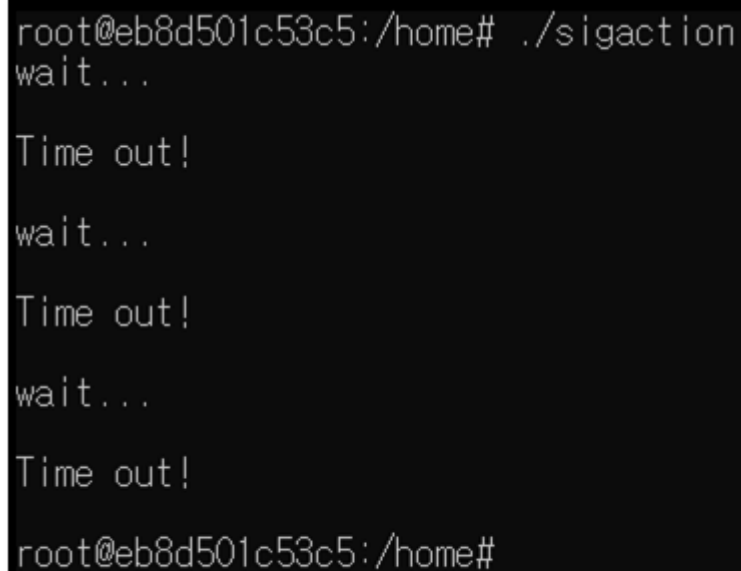
int main(int argc, char * argv[]){
    int i;
```

```

struct sigaction act;
act.sa_handler = timeout;
sigemptyset(&act.sa_mask);
act.sa_flags = 0;
sigaction(SIGALRM, &act, 0);
alarm(2);

for (i=0; i<3; i++){
    puts("wait...\n");
    sleep(100);
}
return 0;
}

```



```

root@eb8d501c53c5:/home# ./sigaction
wait...
Time out!
wait...
Time out!
wait...
Time out!
root@eb8d501c53c5:/home#

```

Example7

remove_zombie.c

```

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

void read_childproc(int sig){
    int status;
    pid_t id = waitpid(-1, &status, WNOHANG);
}

```

```

        if(WIFEXITED(status)){
            printf("Removed proc id : %d\n",id);
            printf("Child send : %d\n", WEXITSTATUS(status));
        }
    }

int main(int argc, char *argv[]){
    pid_t pid;
    struct sigaction act;
    act.sa_handler = read_childproc;
    sigemptyset(&act.sa_mask);
    act.sa_flags=0;
    sigaction(SIGCHLD, &act, 0);
    pid = fork();
    if(pid==0){
        puts("HI! I'm child process");
        sleep(10);
        return 12;
    }
    else{
        printf("Child proc id : %d\n",pid);
        pid=fork();
        if(pid==0){
            puts("Hi! I'm child process\n");
            sleep(10);
            exit(24);
        }
        else{
            int i;
            printf("Child proc id : %d\n",pid);
            for(i=0;i<5;i++){
                puts("wait...");
                sleep(5);
            }
        }
    }
    return 0;
}

```

```

root@eb8d501c53c5:/home# ./remove_zombie
Child proc id : 190
Hi! I'm child process
Child proc id : 191
wait...
Hi! I'm child process

wait...
Removed proc id : 190
Child send : 12
wait...
Removed proc id : 191
Child send : 24
wait...
wait...
root@eb8d501c53c5:/home#

```

Example8

echo_mpserv.c

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30
void error_handling(char *message);
void read_childproc(int sig);
int main(int argc, char *argv[]){
    int serv_sock, clnt_sock;
    struct sockaddr_in serv_adr, clnt_adr;

    pid_t pid;
    struct sigaction act;
    socklen_t adr_sz;
    int str_len, state;
    char buf[BUF_SIZE];
    if(argc!=2){
        printf("Usage : %s <port>\n", argv[0]);
        exit(1);
    }
    act.sa_handler = read_childproc;

```



```

sigemptyset(&act.sa_mask);
act.sa_flags=0;
state = sigaction(SIGCHLD, &act, 0);

serv_sock = socket(PF_INET, SOCK_STREAM, 0);
memset(&serv_adr, 0, sizeof(serv_adr));
serv_adr.sin_family = AF_INET;
serv_adr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_adr.sin_port = htons(atoi(argv[1]));

if (bind(serv_sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr))==-1)
    error_handling("bind() error");

if (listen(serv_sock,5)==-1)
    error_handling("listen() error");

while(1){
    adr_sz = sizeof(clnt_adr);
    clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_adr, &adr_sz);
    if(clnt_sock==-1)
        continue;
    else
        puts("New client connected...\n");
    pid = fork();
    if(pid==-1){
        close(clnt_sock);
        continue;
    }
    if(pid==0){
        close(serv_sock);
        while((str_len=read(clnt_sock, buf, BUF_SIZE))!=0)
            write(clnt_sock, buf, str_len);
        close(clnt_sock);
        puts("client disconnected...");
        return 0;
    }
    else
        close(clnt_sock);
}
close(serv_sock);
return 0;
}

void read_childproc(int sig){
    pid_t pid;
    int status;
    pid=waitpid(-1, &status, WNOHANG);
    printf("removed proc id: %d\n",pid);
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

```

echo_mpclient.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30

void error_handling(char *message);
void read_routine(int sock, char *buf);
void write_routine(int sock, char *buf);

int main(int argc, char *argv[]){
    int sock;
    pid_t pid;
    char buf[BUF_SIZE];
    struct sockaddr_in serv_addr;

    if(argc!=3){
        printf("Usage : %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    sock = socket(PF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));

    if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))== -1)
        error_handling("connect() error!");

    pid = fork();
    if(pid==0)
        write_routine(sock, buf);
    else
        read_routine(sock, buf);
    close(sock);
    return 0;
}

void read_routine(int sock, char *buf){
    while(1){
        int str_len = read(sock, buf, BUF_SIZE);
        if(str_len==0)
            return ;
        buf[str_len]=0;
        printf("Message from server : %s\n", buf);
    }
}

void write_routine(int sock, char *buf){
    while(1){
```

```

        fgets(buf, BUF_SIZE, stdin);
        if(!strcmp(buf, "q\n") || !strcmp(buf, "Q\n")){
            shutdown(sock, SHUT_WR);
            return ;
        }
        write(sock, buf, strlen(buf));
    }
}

void error_handling(char *message){
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

```

```

root@eb8d501c53c5: /home
root@eb8d501c53c5:/home# ./echo_mpserv 9111
New client connected...
New client connected...
New client connected...
client disconnected...
removed proc id: 258
client disconnected...
removed proc id: 261
client disconnected...
removed proc id: 275

root@eb8d501c53c5:/home# ./echo_mpclient 127.0.0.1 9111
Connected.....
Input message(Q to quit): smooth
Message form server : smooth
Input message(Q to quit): q
root@eb8d501c53c5:/home#

root@eb8d501c53c5:/home# ./echo_mpclient 127.0.0.1 9111
Connected.....
Input message(Q to quit): like
Message form server : like
Input message(Q to quit): q
root@eb8d501c53c5:/home#

root@eb8d501c53c5:/home# ./echo_mpclient 127.0.0.1 9111
Connected.....
Input message(Q to quit): butter
Message form server : butter
Input message(Q to quit): q
root@eb8d501c53c5:/home#

```

Problem1

parent of `if fork()`

child of `if (fork())`

- child of `if (fork()) fork()`
- child of `if (fork() || fork())`
 - child of `if (fork() || fork()) fork()`

the number of processes : 5

output : `"1 1 1 1 1 "`

```
root@eb8d501c53c5:/home# ./problem1
1 1 1 1 1 root@eb8d501c53c5:/home#
```

Problem2

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <sys/wait.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netdb.h>

#define BUF_SIZE 30
void error_handling(char *message);
void read_childproc(int sig);
int main(int argc, char *argv[]){
    int serv_sock, clnt_sock;
    struct sockaddr_in serv_adr, clnt_adr;

    pid_t pid;
    struct sigaction act;
    socklen_t adr_sz;
    int str_len, state;
    char buf[BUF_SIZE];

    int i=0;
    struct hostent *host;
    char message[30];
    char* ipaddr;

    if(argc!=2){
        printf("Usage : %s <port>\n", argv[0]);
        exit(1);
```

```

}
act.sa_handler = read_childproc;
sigemptyset(&act.sa_mask);
act.sa_flags=0;
state = sigaction(SIGCHLD, &act, 0);

serv_sock = socket(PF_INET, SOCK_STREAM, 0);
memset(&serv_adr, 0, sizeof(serv_adr));
serv_adr.sin_family = AF_INET;
serv_adr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_adr.sin_port = htons(atoi(argv[1]));

if (bind(serv_sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr))==-1)
    error_handling("bind() error");

if (listen(serv_sock,5)==-1)
    error_handling("listen() error");

while(1){
    adr_sz = sizeof(clnt_adr);
    clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_adr, &adr_sz);
    if(clnt_sock==-1)
        continue;
    else
        puts("New client connected...\n");
    pid = fork();

    if(pid==-1){
        close(clnt_sock);
        continue;
    }
    if(pid==0){
        close(serv_sock);
        while((str_len=read(clnt_sock, buf, sizeof(buf)))!=0) {
            printf("from client : %s\n",buf);
            for(i=0;i<sizeof(buf);i++){
                if (buf[i]=='\n')
                    buf[i]='\0';
            }
            host = gethostbyname(buf);
            if (!host)
                error_handling("gethost...error");

            for(i=0;host->h_addr_list[i];i++){
                printf("i : %d\n",i);
                ipaddr = inet_ntoa(*(struct in_addr *)host->h_addr_list[i]);
                write(clnt_sock, ipaddr, strlen(ipaddr));
            }
        }
        close(clnt_sock);
        puts("client disconnected...");
        return 0;
    }
    else
        close(clnt_sock);
}
close(serv_sock);

```

```

    return 0;
}

void read_childproc(int sig){
    pid_t pid;
    int status;
    pid=waitpid(-1, &status, WNOHANG);
    printf("removed proc id: %d\n",pid);
}

void error_handling(char *message){
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

```

client.c

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30

void error_handling(char *message);
void read_routine(int sock, char *buf);
void write_routine(int sock, char *buf);

int main(int argc, char *argv[]){
    int sock;
    pid_t pid;
    char buf[BUF_SIZE];
    struct sockaddr_in serv_adr;

    if(argc!=3){
        printf("Usage : %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    sock = socket(PF_INET, SOCK_STREAM, 0);
    memset(&serv_adr, 0, sizeof(serv_adr));
    serv_adr.sin_family = AF_INET;
    serv_adr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_adr.sin_port = htons(atoi(argv[2]));

    if(connect(sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr))== -1)
        error_handling("connect() error!");

    pid = fork();
    if(pid==0)
        write_routine(sock, buf);
    else

```

```

        read_routine(sock, buf);
        close(sock);
        return 0;
    }

void write_routine(int sock, char *buf){
    while(1){
        sleep(3);
        fgets(buf, BUF_SIZE, stdin);
        if(!strcmp(buf, "q\n") || !strcmp(buf, "Q\n")){
            shutdown(sock, SHUT_WR);
            return ;
        }
        write(sock, buf, strlen(buf));
    }
}

void read_routine(int sock, char *buf){
    while(1){
        int str_len = read(sock, buf, BUF_SIZE);
        if(str_len==0)
            return ;
        buf[str_len]=0;
        printf("%s\n", buf);
    }
}

void error_handling(char *message){
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

```

```

root@eb8d501c53c5: /home
root@eb8d501c53c5:/home# ./server 9111
New client connected...
New client connected...
from client : www.naver.com
from client : www.baidu.com
-
root@eb8d501c53c5:/home# ./client 127.0.0.1 9111
www.naver.com
23.60.108.213
root@eb8d501c53c5:/home# ./client 127.0.0.1 9111
www.baidu.com
119.63.197.139
119.63.197.151

```