**3. Homework**

**0) Try ex0 below. Who is the parent of ex0?**

ex0.c:

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

void main(){

int x,y;

x=getpid(); // my pid

y=getppid(); // parent’s pid

printf("PID:%d PPID:%d\n", x, y);

for(;;); // to make this process alive

}

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

$ gcc –o ex0 ex0.c

Run ex0 with &. "&" puts process in the background so that you can issue next command to the shell.

$ ex0&

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Now confirm with "ps –f".

$ ps –f

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**Parent of ex0 is /bin/bash**

Now kill ex0.

$ kill xxxx

,where xxxx is the pid of ex0.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**1) Try ex1 below. Why do we have two hello's? What are the PID of ex1 and ex1's child? Who is the parent of ex1?**

ex1.c:

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

void main(){

int x;

x=fork();

printf(“hello\n”);

for(;;);

}

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

$ gcc –o ex1 ex1.c

$ ex1&

hello

hello

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**Hello 가 2개인 이유는 Fork()를 이용해서 ex1가 2개로 복제되었기 때문.**

$ ps -f

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**PID of ex1: 979, PID of ex1’s child: 980**

**PID of Parent of ex1: 955, /usr/bin/bash**

$ kill xxxx(pid of ex1) yyyy(pid of ex1's child)

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**2) Modify ex1.c such that it prints its own pid and the parent pid. Confirm the result with "ps -f". Who is the parent of ex1? Who is the parent of the parent of ex1? Follow the parent link until you reach PID 1 and show all of them.**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

void main(){

int x;

x=fork();

printf("hello. my pid is %d\n", getpid());

printf("and my parent pid is %d\n", getppid());

for(;;);

}

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Parent of ex1 =>

Owner 990 955 pty0 17:36:31 /cygdrive/c/Users/owner/AppData/Roaming/SPB\_Data/d5/ex1

parent of the parent of ex1 =>

owner 955 954 pty0 17:19:08 /usr/bin/bash

UID PID PPID TTY STIME COMMAND

Owner 991 990 pty0 17:36:31 /cygdrive/c/Users/owner/AppData/Roaming/SPB\_Data/d5/ex1

Owner 990 955 pty0 17:36:31 /cygdrive/c/Users/owner/AppData/Roaming/SPB\_Data/d5/ex1

Owner 955 954 pty0 17:19:08 /usr/bin/bash

Owner 954 1 ? 17:19:08 /usr/bin/mintty

**3) Try below (ex2.c). Which hello is displayed by the parent and which hello is by the child?**

void main(){

int x;

x=fork();

printf(“hello: %d\n”, x);

}

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

$ gcc –o ex2 ex2.c

$ ex2

hello: 22644

hello: 0

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Parent: x의 값이 child의 pid인 프로세스 hello: 1001

Child: x의 값이 0인 프로세스 hello: 0

**4) Try below (ex3.c) and show all ancestor processes of ex3 (parent, parent of parent, etc).**

void main(){

int x;

x=fork();

printf(“hello: %d\n”, x);

for(;;); // make the parent and child alive

}

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

$ gcc –o ex3 ex3.c

$ ex3 &

hello: 22644

hello: 0

$ ps -ef

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

UID PID PPID TTY STIME COMMAND

Owner 1009 1008 pty0 17:49:45 /cygdrive/c/Users/owner/AppData/Roaming/SPB\_Data/d5/ex3

Owner 1008 955 pty0 17:49:45 /cygdrive/c/Users/owner/AppData/Roaming/SPB\_Data/d5/ex3

Owner 955 954 pty0 17:19:08 /usr/bin/bash

Owner 954 1 ? 17:19:08 /usr/bin/minty

**5) Try below (ex4.c). Which message was displayed by the parent and which one by the child?**

void main(){

int x;

x=fork();

if (x==0){

printf(“hello: %d\n”, x);

}else{

printf(“hi: %d \n”, x);

}

}

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

$ gcc –o ex4 ex4.c

$ ex4

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

X의 값이 0이면 hello를 출력, 아니면 hi를 출력하므로,

Parent: hi: 1025

Child: hello: 0

**6) Try below (ex5.c). How many hellos do you see? Explain why you have that many hellos. Draw the process tree.**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

void main(){

int x,y;

x=fork();

y=fork();

printf(“hello: %d %d\n”, x, y);

}

$ gcc –o ex5 ex5.c

$ ex5

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명 텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

Total of 4 hellos, 1번 복제할 때 2개의 프로세스, 한번 더 복제하면 2x2=4, 4개의 프로세스. 그러므로 총 4개의 hello가 나온다.

PROCESS TREE

**ex5(pid=origin)**

라인, 스크린샷, 그래프, 폰트이(가) 표시된 사진

자동 생성된 설명

**ex5(pid=origin)** ex5(pid=1195)

**x = 1195** x = 0



**ex5(pid=origin)** ex5(pid=1196) ex5(pid=1195) ex5(pid=1197)

**x=1195, y=1196** x=1195, y=0 x=0, y=1197 x=0, y=0

**7) Try below (ex6.c). How many hellos do you see? Explain why you have that many hellos.**

void main(){

int x,y;

x=fork();

printf(“hello: %d\n”, x);

y=fork();

printf(“hello: %d\n”, y);

}

$ gcc –o ex6 ex6.c

$ ex6

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명 텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

Total of 6 hellos.

6개가 나온 이유는:

처음에 x = fork()가 ex6을 2개로 복제하고 **각 프로세스의 x값을 출력(2개).**

그리고, y = fork()가 복제된 2개의 ex6을 각각 한번 더 복제하므로 총 4개가 복제. 이때 y의 값을 출력하므로 **총 4개의 프로세스의 y의 값을 출력한다.**

따라서 **x값 2개, y값 4개를 출력하므로 hello의 개수는 총 6개가 된다.**

**8) Try below (ex7.c). When you run ex7, how many processes run at the same time? Which process finishes first and which process finishes last? Show the finishing order of the processes. Run ex7 again and compare the finishing order with that of the first run. If different, explain why.**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <unistd.h>

void main(){

int x, i;

for(i=0;i<5;i++){

x=fork();

if (x==0){ // child

int k;

for(k=0;k<1000;k++){

printf("%d-th child running %d-th iteration\n“, i, k);

fflush(stdout); // to make printf work immediately

usleep(10000); // sleep 10000 microseconds

}

exit(0); // child exits after 1000 iterations

}

}

// now parent

printf(“parent exits\n”);

}

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Child는 exit(0) 을 하기 때문에 프로세스가 끝난다. 따라서 child는 fork()를 쓰지 못한다. 부모만 child를 낳을 수 있다.

1st ex7

텍스트, 스크린샷, 폰트, 책이(가) 표시된 사진

자동 생성된 설명 의류, 패브릭, 패턴, 그레이이(가) 표시된 사진

자동 생성된 설명

5개의 프로세스가 동시에 실행된다.

1st finish: 0-th, 2nd finish: 1-th, 3rd finish: 2-th

4th finish: 4-th

5th finish: 3-th

2nd ex7

텍스트, 책, 메뉴, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1st finish: 0-th, 2nd finish: 1-th, 3rd finish: 2-th

4th finish: 3-th

5th finish: 4-th

끝나는 순서가 달라지는 이유는 스케줄러가 fork() 실행 후,

parent를 먼저 실행하거나(새로운 child 복제)

child를 먼저 실행할 수 있기 때문이다(1000번 iteration 실행).

누가 먼저 실행되는지는 각 ex7 실행 경우에 따라 다르다.

**9) If you delete "exit(0)" in ex7.c, how many processes will be created? Confirm your answer by modifying the code such that each process displays its own pid. You may want to decrease the size of inner loop (k loop) to “k<10” to shorten the run time.**

총 31( = 1+2+4+8+16)개의 프로세스가 생겨날 것이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진

자동 생성된 설명

**Child pid: 1617 ~ 1647(총 31개)**

**10) Write a program that creates n child processes where n is specified by the user. Let each process prints some message when it has finished.**

$ ex1

how many child processes?

5

child 1 finished

child 2 finished

child 4 finished

child 3 finished

child 5 finished

parent finished

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**11) Repeat Problem 10, but let the processes end in the reverse order in their creation time as shown below. Use sleep() function to force the order.**

$ ex1

how many child processes?

5

parent finished

child 5 finished

child 4 finished

child 3 finished

child 2 finished

child 1 finished

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**12) Write a program that the parent writes some 1-digit number x into a file and the child reads it and prints x + 1. Assume x is less than 10. Check this file with xxd if it has the number written by the parent.**

$ ex1

parent writes 7

child prints 8

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명