운영체제 Lab03 (fork + exec) 과제

201601692 김재희

1. **onetwo.c**
2. 소스코드

#include *<stdio.h>*

#include *<unistd.h>*

int main() {

int pid;

printf("--------------**\n**학번: 201601692**\n**이름: 김재희**\n**--------------**\n**");

printf("pid=%d One**\n**", getpid());

pid = fork();

printf("pid=%d Two**\n**", getpid());

**return** 0;

}

1. 실행결과



1. **forkexeclp.c**
2. 소스코드

#include *<stdio.h>*

#include *<stdlib.h>*

#include *<unistd.h>*

#include *<sys/wait.h>*

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

{

int pid;

printf("--------------**\n**학번: 201601692**\n**이름: 김재희**\n**--------------**\n**");

*/\* 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", "-l", NULL);

} **else** { */\* parent process \*/*

wait(NULL);

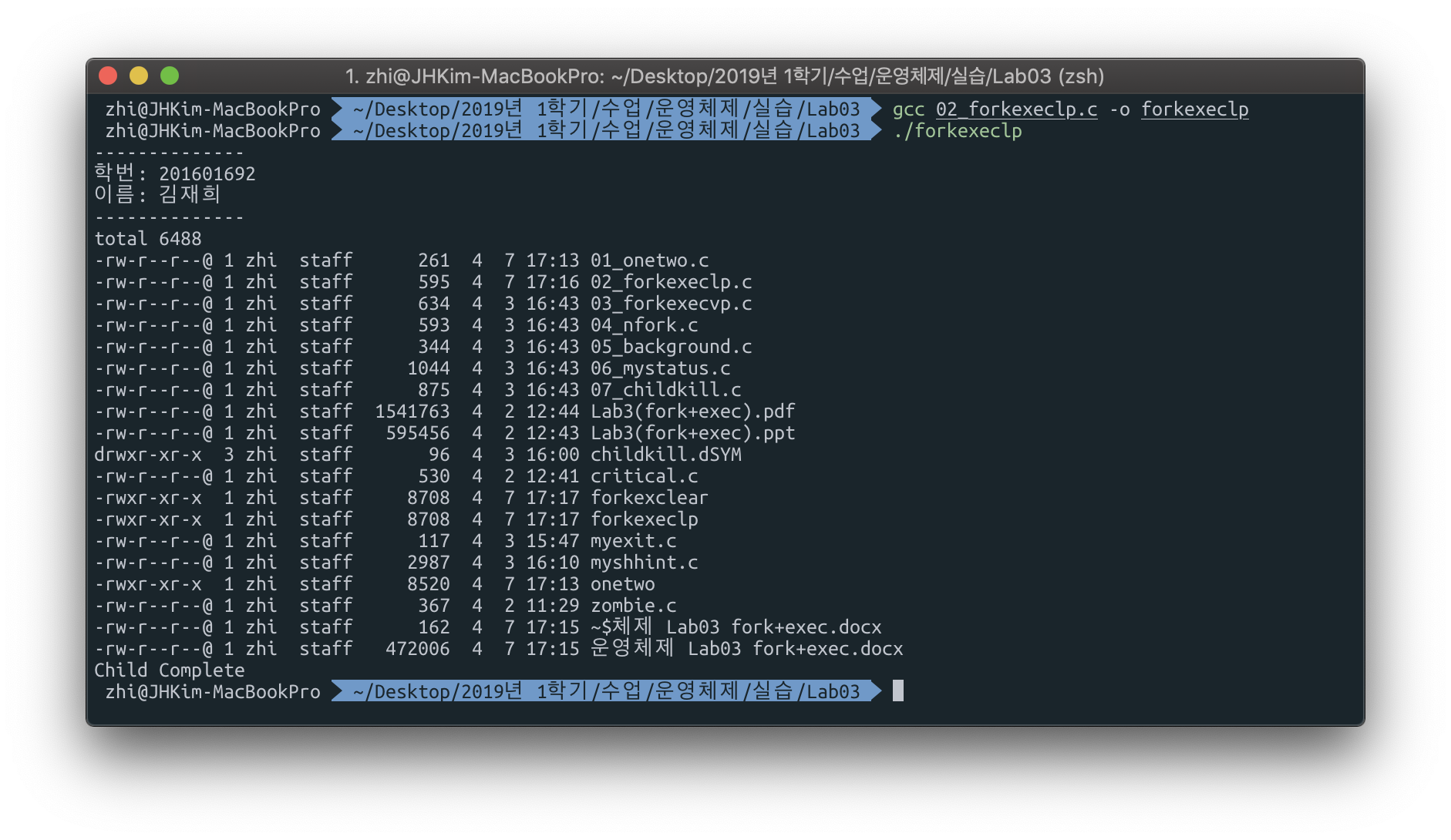
printf("Child Complete**\n**");

exit(0);

}

}

1. 실행결과



1. **forkexecvp.c**
2. 소스코드

#include *<stdio.h>*

#include *<stdlib.h>*

#include *<unistd.h>*

#include *<sys/wait.h>*

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

{

int pid;

char \*av[3];

av[0]="ls";

av[1]="-l";

av[2]=(char \*)0;

printf("--------------**\n**학번: 201601692**\n**이름: 김재희**\n**--------------**\n**");

*/\* fork another process \*/*

pid = fork();

**if**(pid < 0) { */\* error occurred \*/*

fprintf(stderr, "Fork Failed");

exit(-1);

} **else** **if** (pid == 0) { */\* child process \*/*

execv("/bin/ls", av);

} **else** { */\* parent process \*/*

wait(NULL);

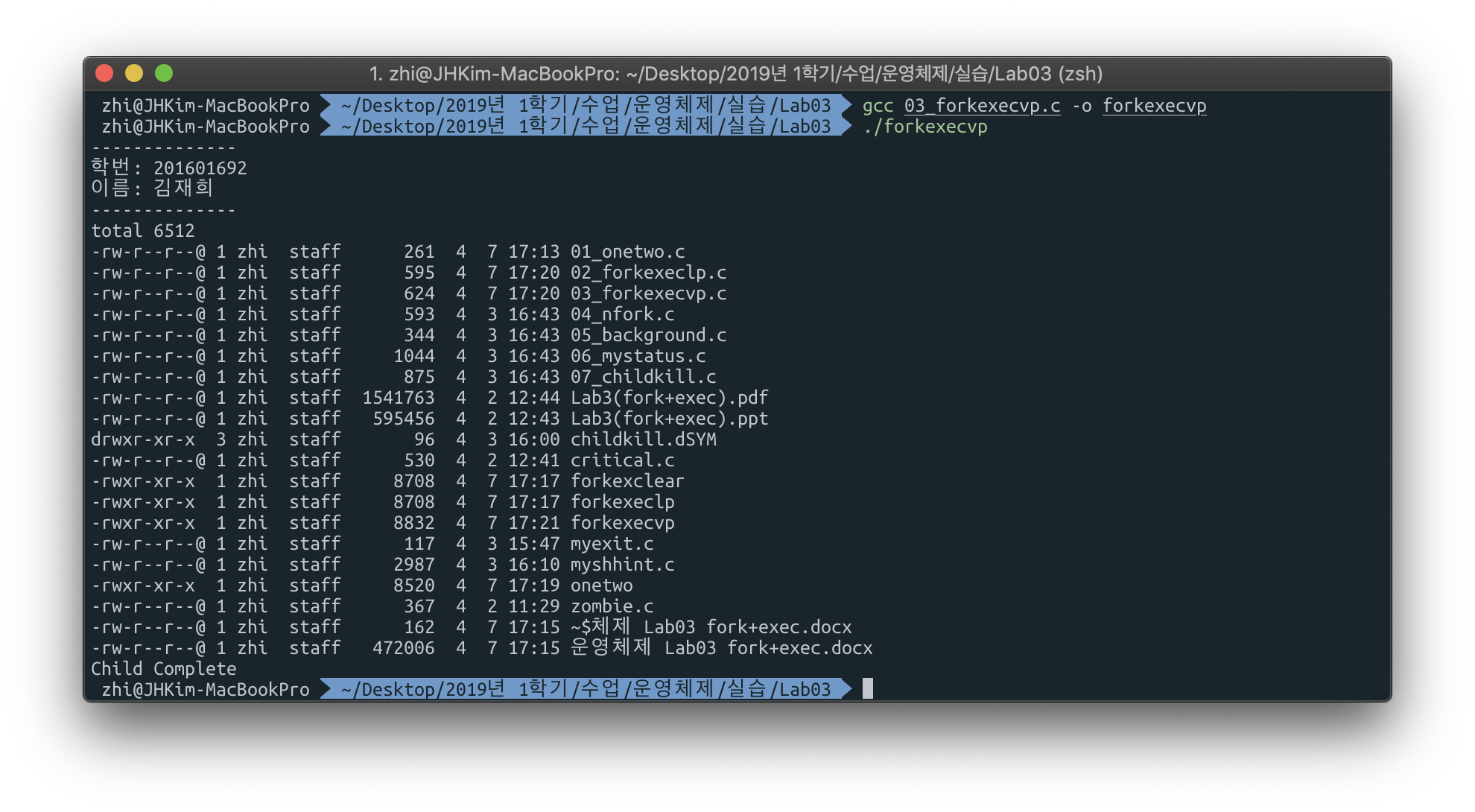
printf("Child Complete**\n**");

exit(0);

}

}

1. 실행결과



1. **nfork.c**
2. 소스코드

#include *<stdio.h>*

#include *<stdlib.h>*

#include *<unistd.h>*

int main()

{

int pid1, pid2;

printf("--------------**\n**학번: 201601692**\n**이름: 김재희**\n**--------------**\n**");

printf("[Parent] : Hello, world ! pid=%d ppid=%d**\n**", getpid(), getppid());

**if** ((pid1=fork()) == 0) {

printf("[Child 1] : Hello, world ! pid=%d ppid=%d**\n**", getpid(),

getppid());

exit(0); */\* ① \*/*

}

**if** ((pid2=fork()) == 0) {

printf("[Child 2] : Hello, world ! pid=%d ppid=%d**\n**", getpid(),

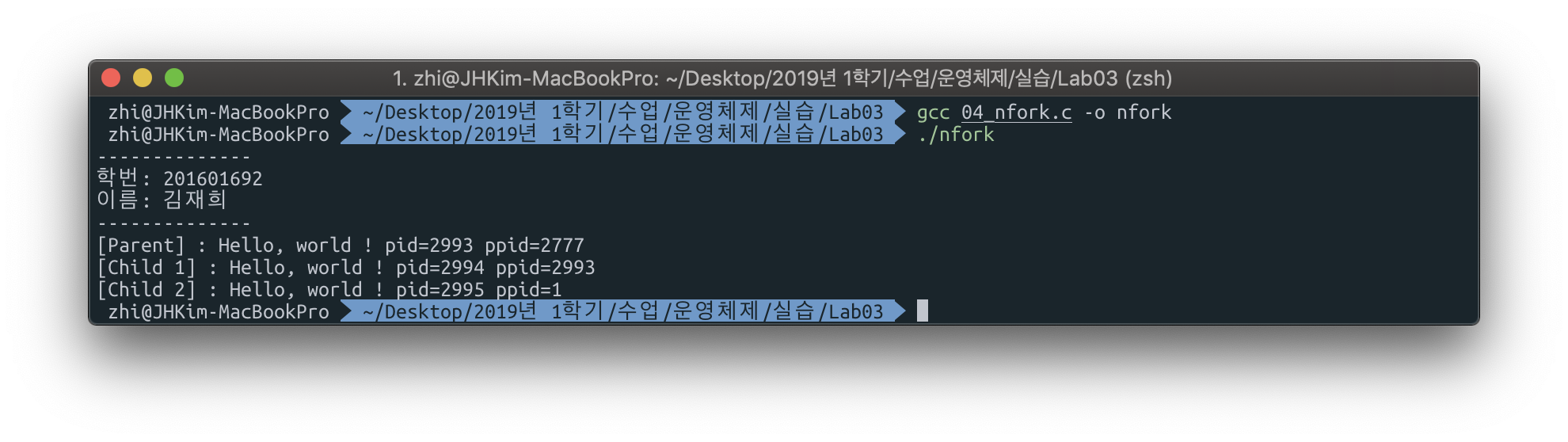
getppid());

exit(0); */\* ② \*/*

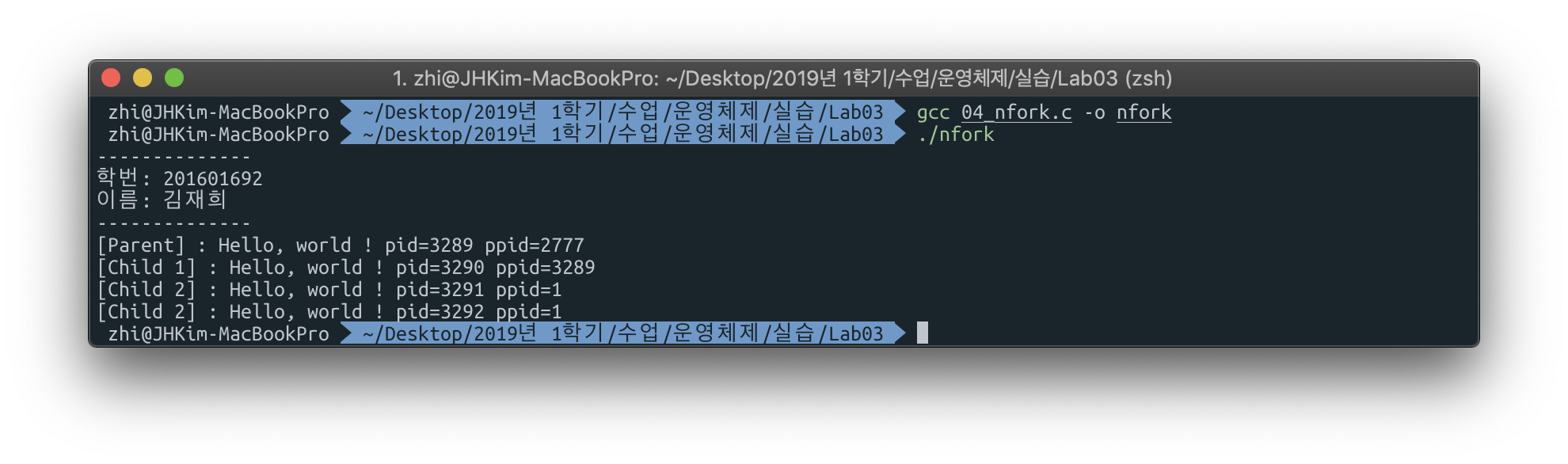
}

}

1. 실행결과
2. ① 과 ② 둘다 있는 것, ②만 없앤 것



1. ①만 없앤 것, ① 과 ② 둘다 없앤 것



1. **background.c**
2. 소스코드

#include *<stdio.h>*

#include *<unistd.h>*

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

printf("--------------**\n**학번: 201601692**\n**이름: 김재희**\n**--------------**\n**");

**if** (fork () == 0) */\* Child \*/*

{

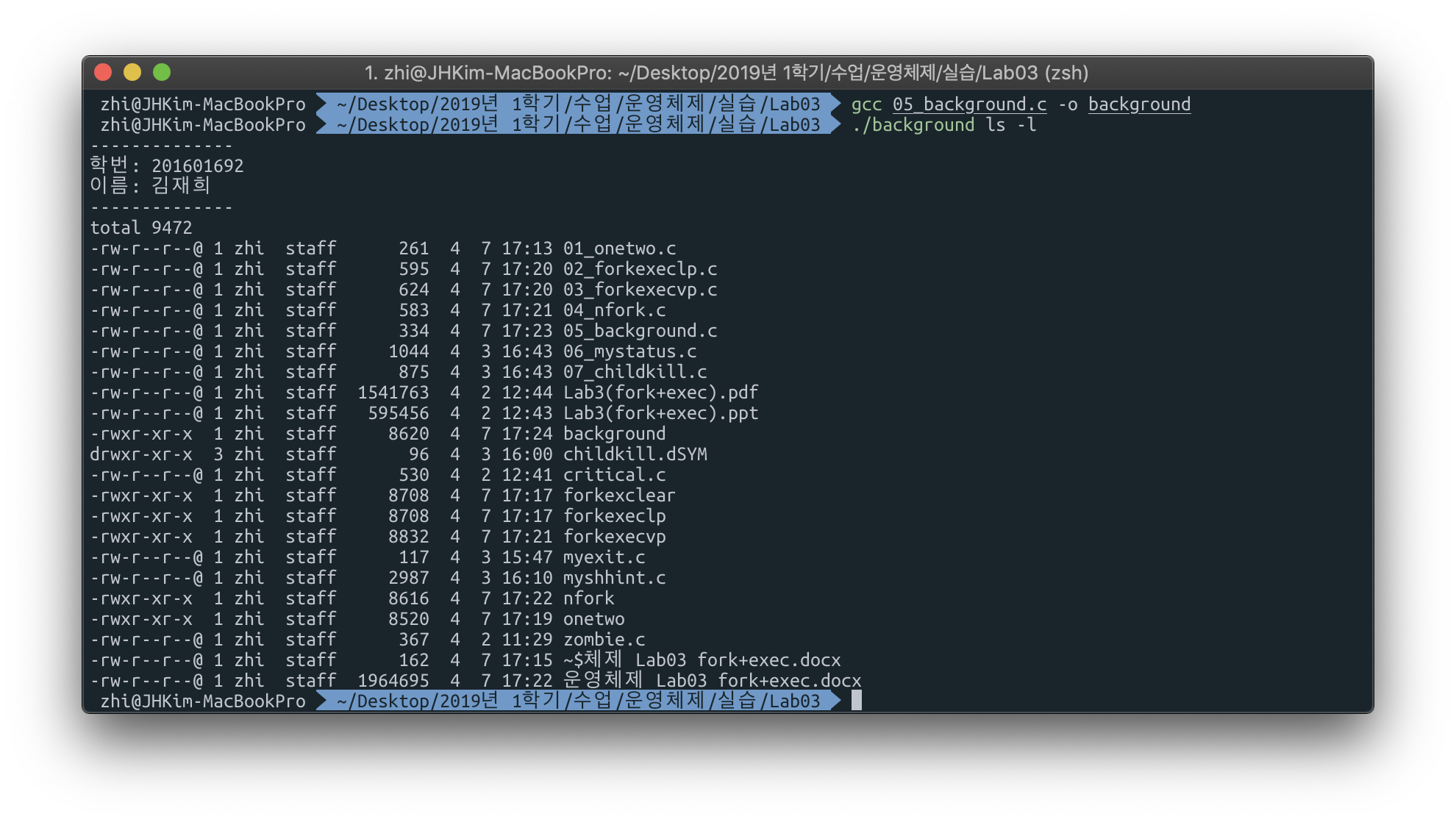
execvp (argv[1], &argv[1]); */\* Execute other program \*/*

fprintf (stderr, "Could not execute %s**\n**", argv[1]);

}

}

1. 실행결과



1. **mystatus.c**
2. 소스코드

#include *<stdio.h>*

#include *<stdlib.h>*

#include *<unistd.h>*

#include *<sys/wait.h>*

int main ()

{

int pid, status, childPid;

printf("-------------**\n**학번: 201601692**\n**이름: 김재희**\n**--------------**\n**");

printf ("I'm the parent process and my PID is %d**\n**", getpid ());

pid = fork (); */\* Duplicate \*/*

**if** (pid != 0) */\* Branch based on return value from fork () \*/*

{

printf ("I'm the parent process with PID %d and PPID %d**\n**", getpid (), getppid ());

childPid = wait (&status); */\* Wait for a child to terminate. \*/*

printf ("A child with PID %d, terminated with exit code high: %d, low: %d**\n**", childPid, status >> 8,

status & 0xFF); */\* Linux \*/*

}

**else**

{

printf ("I'm the child process with PID %d and PPID %d**\n**", getpid (), getppid ());

*/\* execlp ("ls", "ls", "-li", (char \*)0); \*/*

*/\* execlp ("./myexit", "./myexit", (char \*)0); \*/*

exit (42); */\* Exit with a silly number \*/*

}

printf ("PID %d terminates**\n**", getpid () );

}

1. 실행결과



1. **childkill.c**
2. 소스코드

#include *<stdio.h>*

#include *<stdlib.h>*

#include *<unistd.h>*

#include *<sys/wait.h>*

#include *<signal.h>*

*/\* 명령줄 인수로 받은 명령을 실행시킨다 \*/*

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

int child, pid, status;

printf("--------------**\n**학번: 201601692**\n**이름: 김재희**\n**--------------**\n**");

pid = fork( );

**if** (pid == 0) { *// 자식 프로세스*

execvp(argv[1], &argv[1]);

fprintf(stderr, "%s:실행 불가**\n**",argv[1]);

} **else** { *// 부모 프로세스*

printf("SIGINT=%d, SIGKILL=%d, SIGTERM=%d**\n**", SIGINT, SIGKILL, SIGTERM);

kill(pid, SIGINT); */\* SIGKILL과 SIGTERM도 확인 \*/*

child = wait(&status);

printf("[parent: %d] 자식 프로세스 %d 종료 **\n**", getpid(), pid);

printf("[child : %d] 종료 코드 상위: %d 하위: %d **\n**", child, status >> 8, status & 0xFF);

}

}

1. 실행결과

