
Project01 Wiki & Comment

Implementation of system call `getgid()`
and user program



제출일	2024.03.26	전공	컴퓨터소프트웨어학부
과목	운영체제	학번	2020089007
담당교수	강수용	이름	김형민

목차

1. Design	3
1.1 User program: project01	3
1.2 System call: getgid()	3
1.2.1. Kernel side	3
1.2.2. User side	3
2. Implement	4
2.1 User program: project01	4
2.1.1. #include 지시문	4
2.1.2. main() 함수	4
2.2 System call: getgid()	4
2.2.1. 전체적인 추가 요소	4
2.2.2. 전체적인 흐름	4
2.2.3. sys_getgid() 함수 구현 세부사항	4
3. Result	5
4. Trouble shooting	5

1. Design

1.1 User program: project01

명세에서 제공한 예시에 따르면 유저 프로그램은 다음과 같은 순서로 실행되어야 한다.

- A. 콘솔에 "project01"을 입력하여 유저 프로그램을 실행한다.
- B. `printf()` 함수를 통해 나의 학번, 현 프로세스의 pid, 조부모 프로세스의 pid를 출력한다.
- C. 프로세스를 정상적으로 종료한다.

위 기능을 구현하기 위해서는 다음 4가지의 함수가 필요하다.

- A. 출력을 위한 `printf()` 함수
- B. 현 프로세스의 pid를 얻기 위한 `getpid()` 함수
- C. 조부모 프로세스의 pid를 얻기 위한 `getppid()` 함수
- D. 프로세스를 종료하기 위한 `exit()` 함수

`printf()` 함수를 총 3번 사용한다. 이때 표준출력을 의미하는 파일 디스크립터 1을 첫 번째 인자로 넘겨준다. 순서대로 나의 학번, 현 프로세스의 pid, 조부모 프로세스의 pid를 `printf()`를 이용해 출력한다. 마지막으로 `exit()`를 호출해 프로세스를 정상 종료한다.

Makefile에 소스 파일을 추가해 컴파일 및 링킹이 이루어지도록 한다.

1.2 System call: `getppid()`

시스템 콜을 추가하기 위해서는 크게 커널 측면과 유저 측면, 양쪽 모두를 고려해야 한다.

1.2.1. Kernel side

- A. `sys_getppid()` 함수를 `syscall.c` 파일에 등록한다.
- B. `sys_getppid()` 함수의 등록을 위해 `syscall.h` 파일에 새로운 매크로를 추가한다.
- C. `sys_getppid()` 함수를 `sysproc.c` 파일에 구현한다.

일반적으로 시스템 콜을 추가할 때는 시스템 콜 함수를 정의하고, 인자 전처리를 위해 그것을 wrapper 함수로 감싸는 방식으로 구현한다. 그러나 `getppid()` 함수의 경우 인자 전처리가 요구되지 않으므로 함수 정의가 불필요하다. 따라서 wrapper 함수에서 직접 조부모 프로세스의 pid를 반환하는 형태로 구현한다.

1.2.2. User side

- A. 유저가 사용가능하도록 `user.h` 파일에 `getppid()` 함수를 선언한다.
- B. `usys.S` 파일에 전처리를 위한 새로운 매크로를 추가한다.

위에 제시한 절차를 따라 커널과 유저 측면을 모두 고려하여 시스템 콜 `getppid()`를 구현한다.

2. Implement

2.1 User program: project01

2.1.1. #include 지시문

types.h, stat.h 그리고 user.h, 이 3가지 파일을 언급한 순서에 맞게 가져와야 한다. 순서를 지키지 않으면 컴파일 에러가 발생할 수 있다. user.h 파일을 include하면서 printf(), getpid(), getgid() 등 프로그램 실행에 필요한 함수들을 사용할 수 있게 된다.

2.1.2. main() 함수

main() 함수는 나의 학번, 현 프로세스의 pid, 조부모 프로세스의 pgid를 차례대로 출력한다. 이때 표준 출력을 위해 printf() 함수를 사용하는데, file descriptor를 나타내는 첫 번째 인자로 stdout을 의미하는 1을 넘겨줘야 한다. 두 번째 출력에는 getpid() 함수를 사용하고, 세 번째 출력에는 getgid() 함수를 사용한다. 세 번의 출력이 모두 완료되었으면 exit() 함수를 호출해 프로그램을 정상 종료한다.

2.2 System call: getgid()

2.2.1. 전체적인 추가 요소

시스템 콜 구현을 위해 총 5가지를 추가하였다.

1. usys.S : 시스템 콜 호출을 위한 SYSCALL(getgid) 매크로 정의
2. syscall.h : SYS_getgid에 23을 할당하는 매크로 추가
3. syscall.c : sys_getgid() 함수 선언 및 등록
4. sysproc.c : sys_getgid() 함수 구현
5. user.h : 유저의 사용을 위한 getgid() 함수 선언

2.2.2. 전체적인 흐름

"user.h"에 선언된 함수를 이용해 유저 프로그램에서 getgid() 함수 호출 시 "usys.S"에 정의된 매크로를 통해 "syscall.h"에서 SYS_getgid가 나타내는 값 23이 eax 레지스터에 저장된다. 그리고 시스템 콜을 뜻하는 코드 64의 인터럽트를 생성한다. 그 후 "syscall.c"에 정의된 syscall() 함수가 호출되고, 우리가 앞서 sys_getgid() 함수를 등록해놓았기 때문에 "sysproc.c"에 구현한 sys_getgid() 함수가 실행된다. 이 함수는 조부모 프로세스의 pid를 반환한다.

2.2.3. sys_getgid() 함수 구현 세부사항

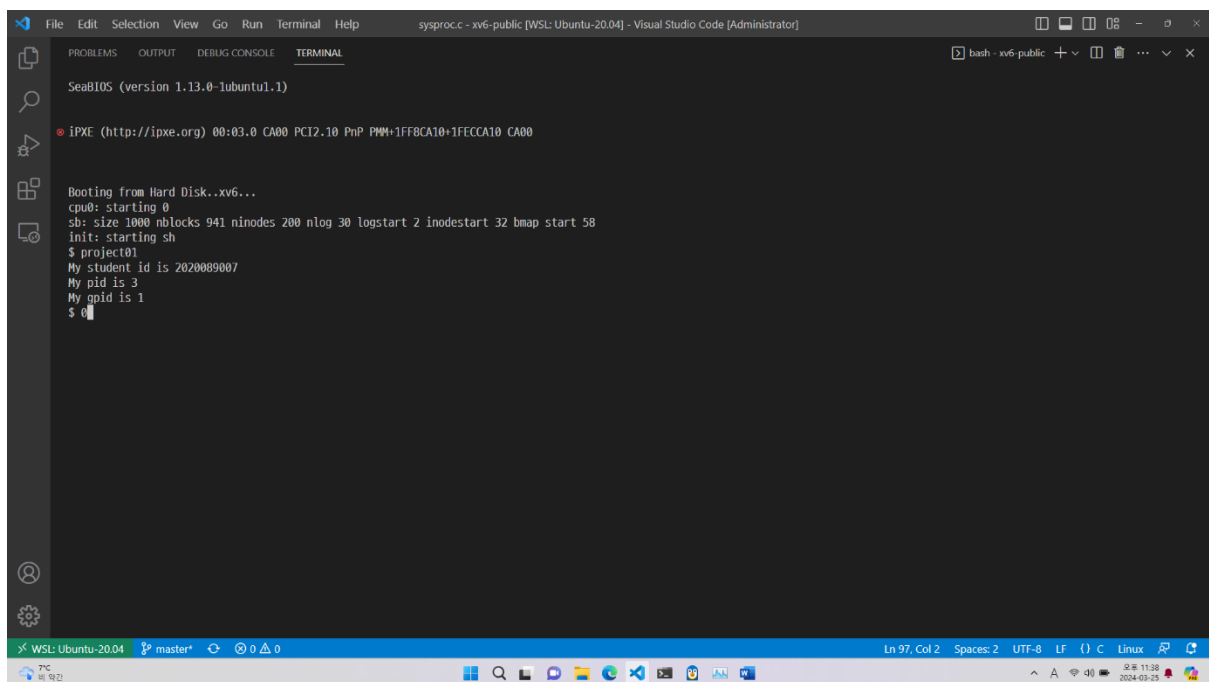
인자 전처리가 요구되지 않기 때문에 단순히 "myproc()->parent->parent->pid"를 통해 조부모 프로세스의 pid를 얻어 반환한다.

3. Result

세부적인 실행 순서는 다음과 같다.

1. "make", "make fs.img"를 콘솔에 입력해 유저 프로그램과 커널을 컴파일한다.
2. "./bootxv6.sh"를 콘솔에 입력해 xv6를 실행한다.
3. "project01"을 xv6 콘솔에 입력해 유저 프로그램을 실행한다.
4. 차례대로 학번, 현 프로세스의 pid, 조부모 프로세스의 pid를 출력한다.
5. 프로세스를 종료한다.

다음은 3번부터 5번까지의 실행 결과를 나타내는 사진이다.



```
SeabIOS (version 1.13.0-1ubuntu1.1)
iPXE (http://ipxe.org) 00:03:0 CA00 PCI2.10 PnP PPM+1FF8CA10+1FECCA10 CA00

Booting from Hard Disk..xv6...
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ project01
My student id is 2020089007
My pid is 3
My ppid is 1
$ exit
```

4. Trobule shooting

유저 프로그램을 실행하는 과정에서 출력은 정상적으로 완료되었음에도 프로세스가 비정상적으로 종료되는 문제를 겪었다. 유저 프로그램의 마지막에서 `exit()`를 해주지 않은 것이 원인이었다. 기존의 윈도우 및 우분투 환경에서 하던 것처럼 관성적으로 하다보니 생긴 문제였다. 이를 깨닫고 마지막에 `exit()` 시스템 콜을 호출해주니 프로세스가 정상적으로 종료되었고, 문제를 해결할 수 있었다.