

# Project01 wiki

2020081958 송재휘

## 1. Design

우선 과제에서 요구하는 `getgpid()`라는 시스템콜은 조부모의 프로세스 id를 반환해야 한다. 이를 구현하기 위해 새로운 파일을 만들기 보다는 `sys_getpid()` 시스템콜이 있는 `sysproc.c` 파일에 `sys_getpid()` 시스템콜을 구현할 계획이다. 이후, `syscall.h`과 `syscall.c`에 `sys_getpid` 시스템콜을 차례대로 등록할 것이다. 그 다음으로는 구현한 시스템콜이 유저 프로그램에서 사용이 가능하도록 `user.h`와 `usys.S`에 `getgpid()`를 등록할 계획이다. 이렇게 되면 시스템콜 등록이 완료될 것이기 때문에 `project01`이라는 이름의 유저 프로그램을 만들어 과제의 예시와 같은 실행 결과를 띄우게 할 것이다.

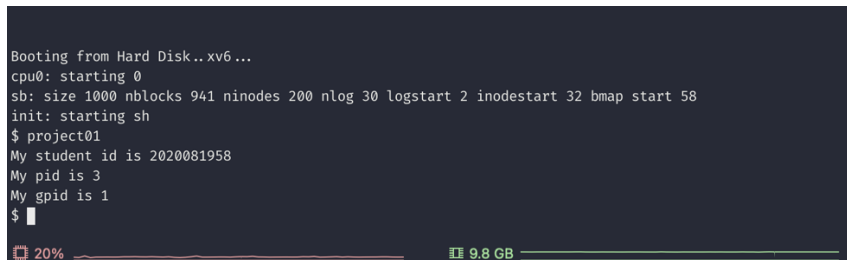
## 2. Implementation

이번 과제는 앞서 언급한 Design의 flow를 따라서 시스템콜을 구현하였다. 첫번째로, `sysproc.c` 파일에 `int sys_getpid(void)`를 구현했는데, 과제 실행 결과의 예시로 나와있는 student id와 현재 pid를 출력하기 위해 해당 함수 안에서 `cprintf`를 통해 바로 나의 학번과 현재 pid를 출력해주었다. 이때 현재 pid를 출력하기 위해 이미 구현되어 있는 시스템콜인 `sys_getpid()`를 실행했다. 마지막으로 `return myproc()->parent->parent->pid;`를 통해 조부모 프로세스의 pid를 반환해주었다. 조부모 프로세스 pid의 출력은 유저 프로그램에서 처리할 예정이다. 그 다음 과정으로 `syscall.h` 파일에 `SYS_getpid`를 등록해주었는데, 이전 실습 과정에서 22번의 시스템콜을 만들었으므로, 23번에 해당 시스템콜을 등록했다. 그 코드는 다음과 같다. (`#define SYS_getpid 23`) 이후, `syscall.c` 파일에 `SYS_getpid`를 등록하기 위해 `extern int sys_getpid(void);`와 `[SYS_getpid] sys_getpid,`를 해당 파일에 추가했다. 여기까지 했다면, 이제 해당 시스템콜을 유저 프로그램에서 사용할 수 있도록 등록하는 과정만 남게 되는데, 이를 위해 먼저 `user.h` 파일에 `int getgpid(void);` 코드를 활용해 `getgpid`를 등록해주었다. 마지막으로 유저 프로그램에서 `getgpid()`를 호출했을 때 이에 해당하는 시스템콜을 실행시키기 위해 `usys.S`에 `SYSCALL(getgpid)`를 등록해주었다. 이렇게 되면 시스템콜 구현은 끝이 났고, 마지막으로 유저 프로그램을 작성하기만 하면 된다. 과제에서 명시된 것처럼 `project01.c` 파일을 만들었고, "`types.h`", "`stat.h`", "`user.h`"를 차례로 `#define` 해주었다. 이후 `int gpid = getgpid();`를 통해 `getgpid()` 시스템콜을 실행시켰고, 해당 시스템콜에서는 나의 학번과, 현재 pid를 제공된 결과 예시와 같게 출력해주고, 조부모 프로세스의 pid를 반환해준다. 이후 `printf(1, "My gpid is %d\n", gpid);`를 통해 조부모 프로세스의 pid를 출력해주고 `exit()`으로 유저 프로그램을 종료하도록 구현했다.

## 3. Result

앞서 설명한 Implementation에 프로그램의 동작원리가 함께 설명되었다. 우선 해당 시스템콜을 컴파일 및 실행하기 위해 Makefile의 UPROGS에 실행파일 이름인 `_project01`을

등록해주었고, EXTRA에 해당 실행파일의 소스코드인 project01.c를 등록해주었다. 이후 make -> make fs.img -> ./bootxv6.sh 명령어를 차례로 실행해 컴파일 및 실행을 했다.. 이렇게 되면 xv6가 구동되고, project01이라는 명령을 입력하게 되면 실행 결과 예시와 같은 결과가 출력된다.

A screenshot of a terminal window showing the boot process of xv6. The text displayed is: "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 2020081958", "My pid is 3", "My gpid is 1", "\$ ". At the bottom, there is a progress bar showing 20% completion and a memory usage indicator showing 9.8 GB.

```
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 2020081958
My pid is 3
My gpid is 1
$
```

프로그램의 동작 원리는 유저 프로그램인 project01을 실행하게 되면 getgid() 명령어가 호출되게 되고, 이를 user.h에 찾는다. 이후 usys.S에서 getgid라는 이름의 system call을 실행하도록 하면, syscall.h에서 해당 시스템콜에 정의된 번호를 찾게 되고, syscall.c에 등록된 대로 sys\_getgid라는 이름의 시스템콜을 실행하게 된다. 이러한 과정을 거쳐 결국, sys\_getgid라는 시스템콜이 구현되어 있는 sysproc.c의 sys\_getgid() 코드가 실행된다.

#### 4. Trouble shooting

xv6라는 운영체제의 코드를 처음 접해보다 보니 해당 운영체제에서 어떤 방식으로 system call을 실행하는지를 파악하는 데에 많은 시간이 걸렸다. 실습 시간에 조교님께서 주신 PPT 자료를 보면서 천천히 그 과정을 따라가보니 xv6에서 어떤 식으로 system call을 수행하는지 조금씩 감을 잡을 수 있었다. 새로운 파일을 만들어 시스템 콜을 구현하는 방식 말고, sys\_getpid()가 구현되어 있는 sysproc.c 파일에 시스템콜을 구현하려고 하다 보니 필요한 단계가 조금 달라 이해하는 데 어려웠지만, 시간을 두고 구조를 잘 들여다보니 오히려 간단하다는 것을 알게 되었다. 새로운 파일을 만들어 시스템콜을 구현할 때와 비교해 새로운 파일을 만들지 않아도 된다는 점과 sysproc.c에 구현을 하게 되면 다른 파일에 해당 함수가 이미 보이기 때문에 defs.h에 따로 추가하지 않아도 된다는 점을 알 수 있었다.