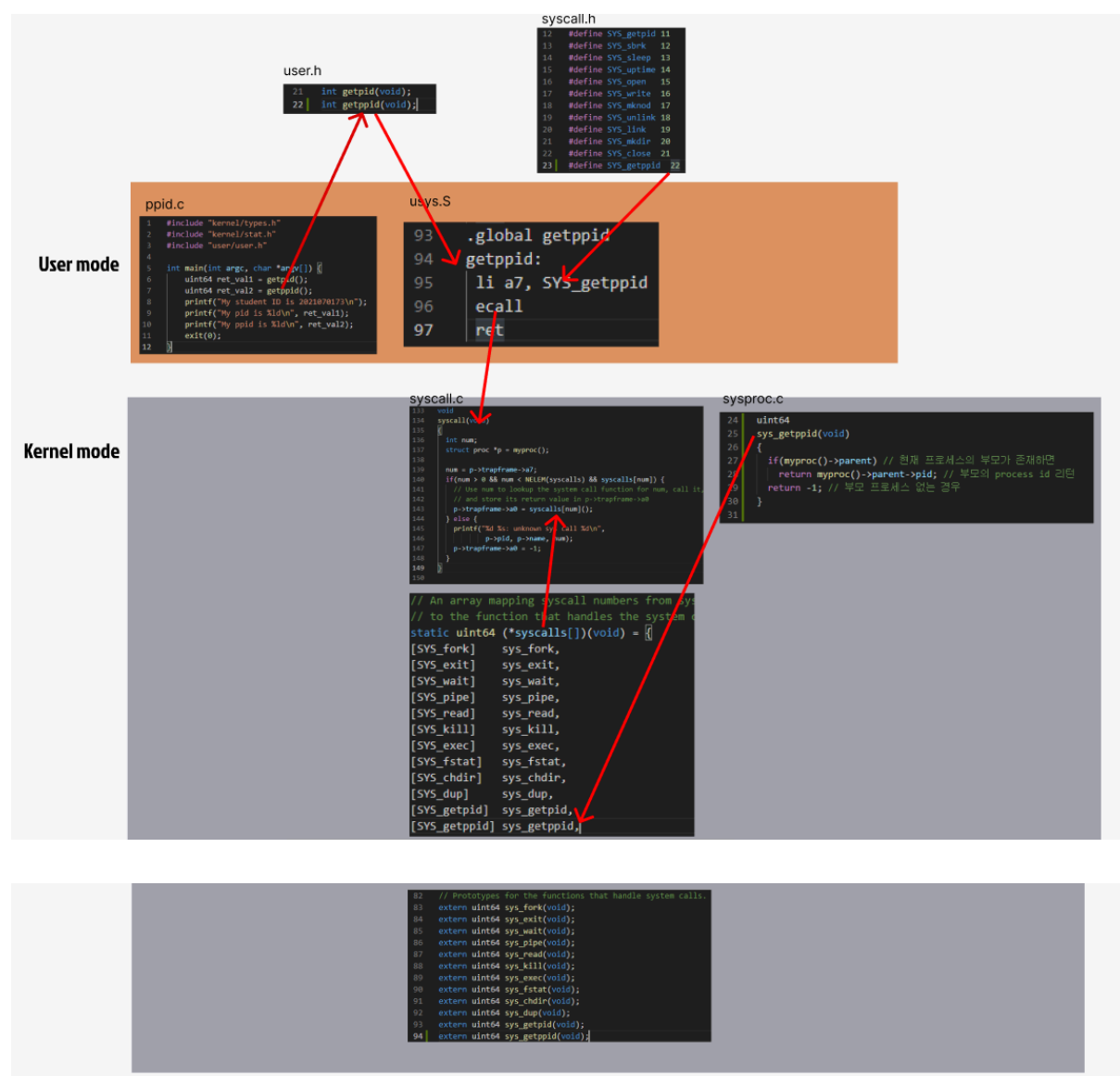


Assignment1 : Implementing a simple system call

Design

현재 프로세스의 부모 프로세스 id를 리턴하는 getppid() 시스템콜을 추가하고, 이를 ppid.c라는 유저 프로그램에서 사용할 수 있도록 만듭니다.

전체 실행 흐름을 보여주는 아키텍처는 다음과 같습니다.



Implementation

구현한 코드와 동작과정은 다음과 같습니다.

1. ppid.c

```
1  #include "kernel/types.h"
2  #include "kernel/stat.h"
3  #include "user/user.h"
4
5  int main(int argc, char *argv[]) {
6      uint64 ret_val1 = getpid();
7      uint64 ret_val2 = getppid();
8      printf("My student ID is 2021070173\n");
9      printf("My pid is %ld\n", ret_val1);
10     printf("My ppid is %ld\n", ret_val2);
11     exit(0);
12 }
```

유저 프로그램을 작성합니다. 이는 시스템콜 호출을 위한 유저 모드의 프로그램입니다.

ppid가 실행되면 getpid()의 유저 모드 함수가 실행됩니다.

(user.h 파일에 getppid() 함수의 프로토타입을 적었습니다.)

```
21  int getpid(void);
22  int getppid(void);
```

2. Makefile 에 운영체제에 포함될 사용자 프로그램들이 정의된 UPROGS 목록에 제가 만든 ppid프로그램을 추가하여 사용할 수 있도록 합니다.

```

125  UPROGS=\
126      $U/_cat\
127      $U/_echo\
128      $U/_forktest\
129      $U/_grep\
130      $U/_init\
131      $U/_kill\
132      $U/_ln\
133      $U/_ls\
134      $U/_mkdir\
135      $U/_rm\
136      $U/_sh\
137      $U/_stressfs\
138      $U/_usertests\
139      $U/_grind\
140      $U/_wc\
141      $U/_zombie\
142      $U/_ppid\

```

3. usys.S

다음의 어셈블리코드를 추가합니다.

```

93  .global getppid
94  getppid:
95      li a7, SYS_getppid
96      ecall
97      ret

```

유저 프로그램에서 getppid함수를 호출하면, 컴파일러는 usys.S의 어셈블리코드로 연결합니다.

- `li a7, SYS_getppid` RISC5 프로세서의 a7 레지스터에 SYS_getppid값을 저장합니다. 이 때 `syscall.h` 에 아래와 같이 SYS_getppid를 22로 정의했으므로 a7레지스터에는 22가 저장됩니다.

```

12  #define SYS_getpid 11
13  #define SYS_sbrk 12
14  #define SYS_sleep 13
15  #define SYS_uptime 14
16  #define SYS_open 15
17  #define SYS_write 16
18  #define SYS_mknod 17
19  #define SYS_unlink 18
20  #define SYS_link 19
21  #define SYS_mkdir 20
22  #define SYS_close 21
23  #define SYS_getppid 22

```

- **ecall**: 사용자모드에서 커널 모드로 변경되는 함수입니다.
 - 커널은 해당하는 시스템콜 핸들러를 실행합니다. (아래의 syscall.c)
- **ret**: 시스템 콜이 완료되면 호출한 함수로 반환합니다.

4. syscall.c

시스템콜 핸들러가 구현되어 있습니다.

```

133 void
134 syscall(void)
135 {
136     int num;
137     struct proc *p = myproc();
138
139     num = p->trapframe->a7;
140     if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
141         // Use num to lookup the system call function for num, call it,
142         // and store its return value in p->trapframe->a0
143         p->trapframe->a0 = syscalls[num]();
144     } else {
145         printf("%d %s: unknown sys call %d\n",
146             p->pid, p->name, num);
147         p->trapframe->a0 = -1;
148     }
149 }
150

```

num에는 a7레지스터 값인 22가 담기고,
 143번째 줄에서 syscall[num]()이 호출되면 아래 로직이 실행되고,
 a0에 **sys_getppid()**의 결과인 부모 프로세스의 id가 담겨 리턴됩니다.

제가 만든 시스템콜 getppid를 등록해주었습니다.

```

// An array mapping syscall numbers from sys
// to the function that handles the system c
static uint64 (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_exit]    sys_exit,
[SYS_wait]    sys_wait,
[SYS_pipe]    sys_pipe,
[SYS_read]    sys_read,
[SYS_kill]    sys_kill,
[SYS_exec]    sys_exec,
[SYS_fstat]   sys_fstat,
[SYS_chdir]   sys_chdir,
[SYS_dup]     sys_dup,
[SYS_getpid]  sys_getpid,
[SYS_getppid] sys_getppid,

```

`syscalls[]` 는 시스템콜을 처리하기 위한 함수 포인터 배열이며,

시스템콜 번호에 해당하는 인덱스 [23]에 시스템 콜을 처리하는 함수의 포인터 `sys_getpid` 가 들어가 있습니다.

즉, `syscalls[num]()` 은 `sys_getpid()` 와 마찬가지로입니다.

```

82 // Prototypes for the functions that handle system calls.
83 extern uint64 sys_fork(void);
84 extern uint64 sys_exit(void);
85 extern uint64 sys_wait(void);
86 extern uint64 sys_pipe(void);
87 extern uint64 sys_read(void);
88 extern uint64 sys_kill(void);
89 extern uint64 sys_exec(void);
90 extern uint64 sys_fstat(void);
91 extern uint64 sys_chdir(void);
92 extern uint64 sys_dup(void);
93 extern uint64 sys_getpid(void);
94 extern uint64 sys_getppid(void);

```

(커널 내의 시스템콜 함수가 사용자 모드에서 호출될 수 있도록 추가)

5. sysproc.c

시스템 콜을 처리하는 함수를 구현한 부분이다.

부모 프로세스의 pid를 구하는 로직이 실행된다.

```

24  uint64
25  sys_getppid(void)
26  {
27      if(myproc()->parent) // 현재 프로세스의 부모가 존재하면
28          return myproc()->parent->pid; // 부모의 process id 리턴
29      return -1; // 부모 프로세스 없는 경우
30  }
31

```

Results

vscode에서 WSL환경을 연동하여 진행하였습니다.

```

sso0711@DESKTOP-IH8REUE:~$ qemu-system-riscv64 --version
QEMU emulator version 8.2.2 (Debian 1:8.2.2+ds-0ubuntu1.6)
Copyright (c) 2003-2023 Fabrice Bellard and the QEMU Project developers
sso0711@DESKTOP-IH8REUE:~$ riscv64-linux-gnu-gcc --version
riscv64-linux-gnu-gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

```

```

sso0711@DESKTOP-IH8REUE:~/assignment-2021070173$ ls
LICENSE Makefile README README.md fs.img kernel mkfs user
sso0711@DESKTOP-IH8REUE:~/assignment-2021070173$ make qemu

```

```

xv6 kernel is booting

init: starting sh
$ ppid
My student ID is 2021070173
My pid is 3
My ppid is 2
$

```

xv6부팅 후 유저 프로그램 ppid를 실행하면,
나의 학번과 현재 프로세스 id, 부모 프로세스 id가 차례로 출력됩니다.

Trouble shooting

1. 내가 만든 유저 프로그램인 ppid.c를 Makefile에 등록해주지 않아 컴파일 되지 않아 생긴 문제였다.

```
xv6 kernel is booting

init: starting sh
$ ppid
\exec ppid failed
$ ppid
exec \ppid failed
```

2. getpid와 getppid 함수의 리턴 타입은 uint64이므로 서식 지정자는 %ld로 적어주어야 했다.

```
iltin-vprintf -I. -fno-stack-protector -fno-pie -no-pie -c -o user/ppid.o user/ppid.c
user/ppid.c: In function 'main':
user/ppid.c:9:24: error: format '%d' expects argument of type 'int', but argument 2 has type 'uint64' {aka 'long unsigned int'} [-Werror=format=]
   9 |     printf("My pid is %d\n", ret_val1);
      |                        ~^      ~~~~~
      |                        |      |
      |                        int  uint64 {aka long unsigned int}
      |                        %ld
user/ppid.c:10:25: error: format '%d' expects argument of type 'int', but argument 2 has type 'uint64' {aka 'long unsigned int'} [-Werror=format=]
  10 |     printf("My ppid is %d\n", ret_val2);
      |                        ~^      ~~~~~
      |                        |      |
      |                        int  uint64 {aka long unsigned int}
      |                        %ld
cc1: all warnings being treated as errors
make: *** [<built-in>: user/ppid.o] Error 1
```

3. 코드변경 후 make clean을 하면 usys.S의 변경사항이 사라져 제대로 실행되지 않았다. Makefile에서 usys.S를 수정하거나 make clean을 하지 말아야한다.

👋 Notion에 오신 것을 환영합니다!