운영체제 Lab 2 보고서

# 분석

exit와 wait가 어떻게 정의가 되어있는지 확인하기 위해 ‘grep exit \*.[chS]’ 명령어를 사용했다.

















defs.h, proc.c, syscall.c, syscall.h, sysproc.c, user.h, usys.S의 7개의 파일에서 정의되어 있고, wait도 똑같이 정의되어 있다.

trap.c에 정의되어 있는 trap 함수를 보면 trap의 번호가 T\_SYSCALL(64, traps.h에 정의되어 있다.)인 경우 호출된 트랩을 현재 프로세스에게 전달하고 syscall함수를 호출한다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

syscall 함수에서는 trap에서 전달받은 트랩프레임의 eax 값을 num으로 가져온다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

eax에는 usys.S에서 “SYS\_ + 시스템 콜의 이름”의 형태로 넣어준 값이 있는데 이 값은 syscall.h에서 선언되어 있다,

폰트, 마젠타, 바이올렛색, 그래픽이(가) 표시된 사진

자동 생성된 설명

exit의 경우 SYS\_exit가 syscall.h에서 2로 정의되어 있으므로 usys.S에서 eax 레지스터에 2값이 들어가고 syscall 함수의 num에 2가 들어간다.

syscalls[num](); 의 함수 포인터에서 2(SYS\_exit)에 대응하는 함수 sys\_exit가 호출된다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

해당 함수의 원형은 위에 extern으로 선언되어 있었고 몸체는 sysproc.c에서 찾을 수 있다.



텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

만약 매개 변수가 있는 함수의 경우 sys\_\*에서 argint 같은 함수로 인수를 넣어주는 것을 확인할 수 있다.

defs.h에서는 void exit(void) 함수의 원형이 선언되어 있는데 주석으로 해당 함수의 몸체는 proc.c에 있다고 적혀 있다.

텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

proc.c에 가보면 수정해야 할 exit 함수의 몸체를 찾을 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

exit 함수가 user.h에 정의되어 있는 것은 유저 프로그램에서 사용하기 위해 정의되어 있는 것이다.

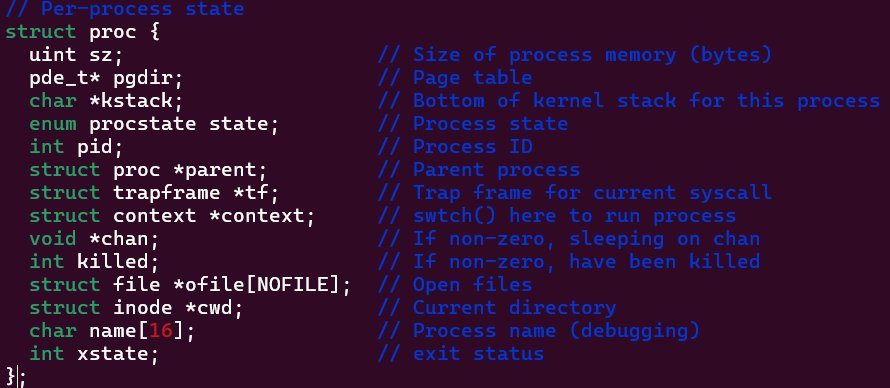
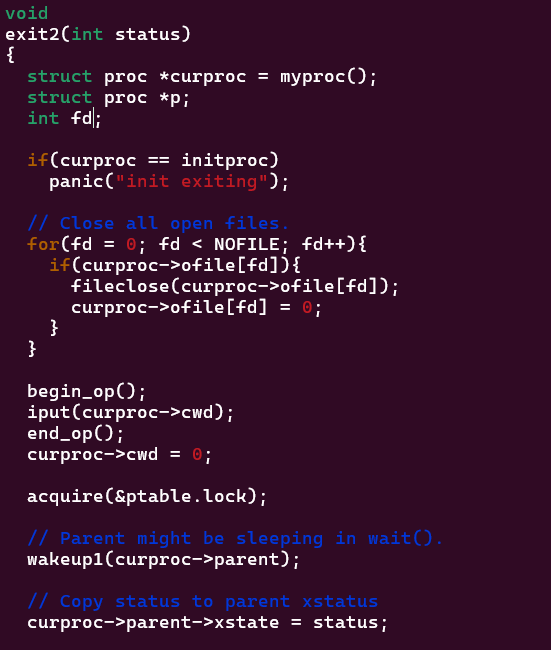
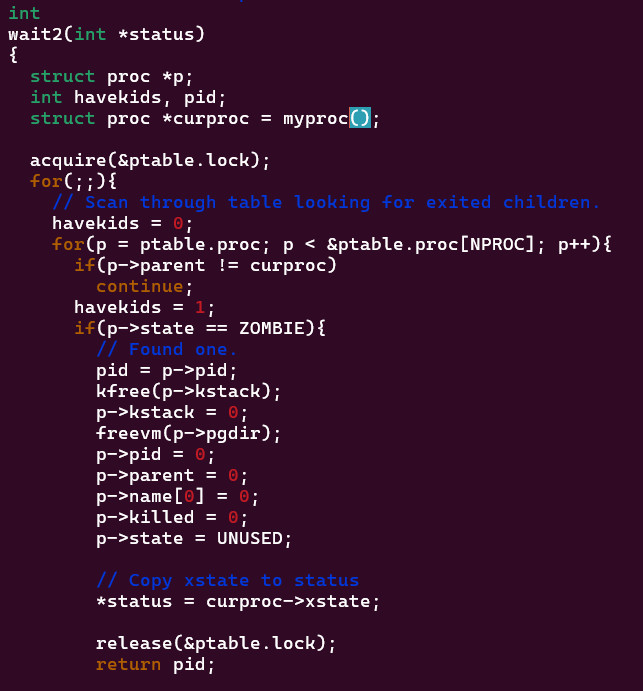
wait 함수도 같은 방법으로 찾을 수 있었다.

우리가 만들려는 wait2와 exit2의 함수는 다음과 같이 동작해야 한다.

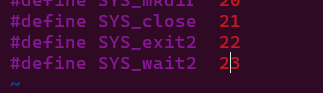
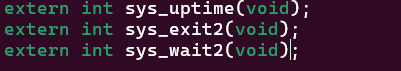
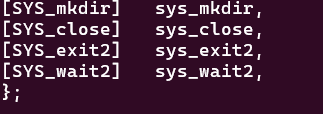
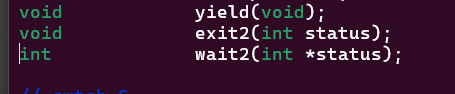
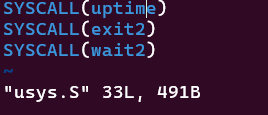
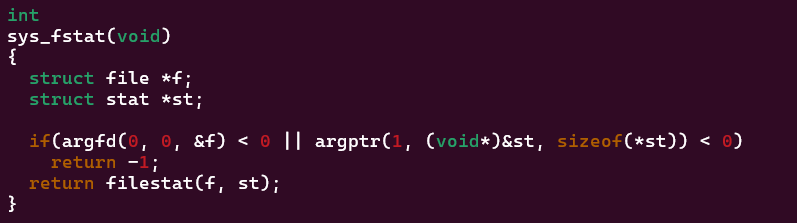
1. exit2에 인수로 status를 전달해야 한다.
2. exit2 내부에서 부모 프로세스의 PCB에 전달받은 status를 넣어주어야 한다.
3. 부모 프로세스는 wait2 함수 내부에서 전달받은 status를 매개변수로 받아온 포인터에 넣어주면 된다.

해당 내용을 바탕으로 다음과 같이 과제를 수행하였다.

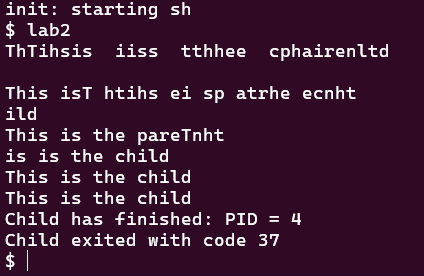
# 수행

1. PCB에 xstate 정보를 추가하기 위해 proc.h에서 proc 구조체의 맨 마지막에 int xstate; 를 추가했다.  
     
   
2. proc.c에서 exit와 wait 함수를 복사하여 각각 exit2와 wait2라고 이름 붙이고, 만들려는 함수에 맞게 매개변수를 넣었다.  
     
   exit2함수에서는 부모 프로세스의 PCB의 xstate값을 매개 변수로 받아온 status의 값을 넣어주기 위해

curproc->parent->xstate = status; 문장을 추가하고   
wait2에서는 자식 프로세스가 끝난 것을 확인하고 자식 프로세스를 종료하는 과정에서 PCB의 xstate의 값을 매개변수로 받은 포인터 변수 status에 넣는 작업을 수행하기위해  
\*status = curproc->xstate; 문장을 추가했다.

1. exit2와 wait2를 시스템 콜로 등록하기 위해 syscall.h에 시스템 콜 번호를 정의했다.
2. syscall.c에서 두 함수를 반환형이 int에 매개변수 없는 함수형으로 선언하고 함수 포인터의 사용을 위해 함수 포인터에도 추가했다.  
     
   
3. defs.h 에다 proc.c에 선언한 함수의 원형을 선언했다.  
   
4. usys.S에서 eax에 값을 넣어 주어야 하므로 SYSCALL()형태로 선언했다.  
   
5. 만든 함수는 매개변수가 있는 함수지만 syscall.c에서는 void를 선언했으므로 exit와 wait와 같이 sysproc.c에 매개변수가 없는 함수에서 직접 만든 함수를 호출할 수 있게 만들었다.  
     
     
     
   exit2의 경우 단순 정수형을 매개변수로 전달하기 때문에 쉽게 만들 수 있었지만 wait2의 경우 포인터를 전달하기 때문에 grep argptr \* 명령어를 통해 비슷한 명령어를 쓰는 함수를 찾아보았고 sysfile.c에서 sys\_fstat을 참고하여 sys\_wait2를 만들었다.  
     
   
6. 시스템 콜을 모두 만들었으니 일반 프로그램에서 해당 함수를 사용할 수 있게 user.h에 함수의 원형을 선언했다.  
   
7. lab2를 명령어로 사용하기 위해 Makefile에 lab2 명령어를 등록했다.  
   
8. lab2.c의 이름으로 파일을 만들었다.  
   

# 결과



“This is the child”와 “This is the parent”가 각각 5번, 3번 출력된 이후 자식 프로세스의 exit\_code 값 37이 부모 PCB로 전달되어 status에 저장되는 것을 확인할 수 있다.