소프트웨어학부 3학년 / 20180285 / 박민준

운영체제 2차과제

개발 환경 : vmware : ubuntu 20.04 :: Linux

수행 과정

1. 시스템콜 등록 :: 새로운 파일 추가하는 방식으로 진행
   1. forknexec.c 생성 및 구현 // 첨부 소스코드 참고.

구성 : int forknexec(const char \*path, const char \*\*args);

int sys\_forknexec(void);

* 1. Makefile : forknexec.o 추가

텍스트, 장치, 측정기이(가) 표시된 사진

자동 생성된 설명

* 1. defs.h : forknexec(const char \*, const char \*); 추가

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. forknexec.c 내에 wrapper 함수인 sys\_forknexec() 추가

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. syscall.h : #define SYS\_forknexec 22 추가

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. syscall.c : extern int sys\_forknexec(void); 및 [SYS\_forknexec] sys\_forknexec, 추가

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. user.h, usys.S 에 등록



텍스트이(가) 표시된 사진

자동 생성된 설명

1. user application 실행 결과

텍스트, 화면, 모니터, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 수정 및 작성한 소스코드에 대한 설명
   1. int forknexec(const char\* path, const char\*\* args);
      1. proc.c 파일 내 구현되어 있는 fork() 함수의 내용과 fork() 에서 호출하는 allocproc 함수를 일부 수정하여 fork에 대한 작업을 수행하고 이를 proc np 변수에 저장하였다. 그리고 np를 runnable 상태로 바꾸어 자식 프로세스를 실행 시키고, 기존의 부모 프로세스는 wait() 시킨 뒤, exec함수의 내용을 가져와 일부 수정한 뒤 이를 수행하도록 하였다. 이때, args 에 대한 오류는 모두 -1을 반환하도록 구성하고, 그 외 오류는 -2를 반환하도록 구현하였다.
   2. int sys\_forknexec(void);
      1. user mode 에서 접근하여 sys\_forknexec 함수를 통해 system call인 forknexec 함수에 접근할 수 있도록 wrapper 함수를 따로 지정해 주었다. 이때, user mode와 kernel mode간 변수 전달이 원활하게 이루어 질 수 있도록 syscall.c 파일 내 구현 되어있는 argstr 함수를 적절히 사용하였다.
2. 수행 중 발생한 문제점 / 해결 방법
   1. User mode, Kernel mode 간 변수 전달
      1. User application인 testforknexec 에서 매개변수로 넣어준 path 와 args가 시스템 콜을 진행하기 위해 Kernel mode로 바뀌면서 변수가 제대로 전달되지 않는 오류가 발생하였다. 이를 해결하기 위해 syscall.c 파일 내에 구현되어 있는 argstr과 fetchstr 함수를 이용하여 system call이 발생하면 호출되는 sys\_forknexec 함수 내에 실행해 주어 path 변수와 args 변수를 kernel mode 내의 forknexec 함수에 전달할 수 있었다.
   2. Const char, char 간 강제 형 변환
      1. Const char \* 로 선언된 매개변수들이 char\* 로 구성되어 있는 여러 함수의 변수들과 충돌하는 문제가 발생 하여 const char \* 형태의 변수들을 (char \*) 강제 형 변환을 통해 오류를 해결 해 주었다.