Os_project01 wiki

2017030055 윤상현

Project01-1

Design & implement 저는 이번 과제인 getppid() 의 구현을 참고하기 위하여 cscope를 사용하였습니다. 우선 저는 과제를 수행하며 제가 만든 유저 프로그램에서 getpid() 함수가 동작하는 것을 통해 제가 실습을 보고 따라하며 추가한 헤더들 어딘가에 getpid함수가 정의되있다고 생각 되어 cscope를 통해 sysproc.c의 위치에 해당 함수가 존재한다는 것을 파악하였습니다.

그리고 해당 함수를 분석하며 현재 프로세스의 id를 반환하는 방식에 대하여 파악하였습니다. 여기서 저는 또 다시 cscope를 활용하여 parent 를 통하여 부모 프로세스에 접근 할 수 있다는 것을 파악하였습니다. 이에 따라 저는 parent-> 를 통하여 부모 프로세스가 가지고 있는 값에 접근하고 process id 값을 getpid와 동일한 방식으로 가져온다면 부모 프로세스의 pid를 가져오는 함수를 구현할 수 있을 것이라고 생각하고 구현하였습니다.

저는 이후 syscall.h와 syscall.c에 제가 만든 함수를 등록하였습니다. 또한 유저 코드에서 참조가 가능한 헤더가 모여있는 파일인 user.h 에 접근하여 getppid()를 추가해주어 유저코드에서 접근할 수 있게 만들었고, usys.S 에 매크로로 등록하였습니다.

이러한 과정을 통해 유저코드가 시스템 콜에 접근할 수 있도록 완성을 한 뒤, 이를 하이레벨 환경에서 실행시킬 유저코드를 작성한 뒤 테스트 해본결과 올바른 값이 도출되었습니다.

유저 코드는 간단하게 메인함수 내부에 prinf 함수를 통하여 user.h로부터 받은 getpid() 와 getppid()를 통해 올바른 pid 와 ppid 값을 출력하는 것으로 완료하였습니다.

Result

실행결과는 명시하신 결과와 같이 첫 입력 때 getpid 값은 3, getppid 값은 2 가 나왔습니다. Getppid는 현재 프로세스의 부모 프로세스의 id 를 리턴하는 것인데, 자식 프로세스는 부모 프로세스에 의해 커널에 요청된 뒤 생성되는 것이기에 이처럼 getppid의 프로세스의 식별값이 getpid의 식별값보다 앞순번으로 나온 것 같습니다.

Trouble

제가 가장 고생했던 부분은 바로 시스템콜에선 눈에 보이지 않는 사소한 요소도 큰 오류로 다가올 수 있다는 기존 하이레벨 코딩과의 차이점 이었습니다. 한가지 예로 제가 makefile을 만들 때, 계속해서 make할 타겟이 없다는 에러를 접하였는데요, 계속해서 문제를 찾아본결과 ₩는 '끝' 을 나타내는 것이기에 그 뒤에 아무것도, 심지어 공백조차 허용이 되지 않는다는 것이였습니다. 공백이 눈에 보이지 않아 문제를 찾는데에만 오랜 시간이 걸렸지만, 기존 모든 것이 자동으로 수정되고 오류의 원인을 명확하게 보여주던 기존의 환경에서 길들여진 안일하게 프로그래밍을 하던 습관에 대한 문제의식을 느끼게 되어 많은 도움이 되었습니다.

Project01-2

저는 과제를 하기에 앞서 xv6, 그리고 시스템콜과 같은 제게 평소에 생소했던 개념들을 이해하기 위해 공부했습니다. 시스템콜은 컴퓨터 시스템이 운영체제의 커널에게 어떠한 것을 요청하는 것으로 새로운 프로세스의 생성과 제거, 하드웨어 관련 서비스 등등에서 시스템콜을 통하여 컴퓨터가 돌아갑니다.

또한 xv6로 운영체제 실습을 진행하고 있었지만 명확하게 xv6가 무엇을 위한 것인지에 대하여

인지하지 못하고 있었는데, 이번 기회에 Xv6는 운영체제 엔지니어들의 교육을 목적으로 개발되었다는 것을 공부하였습니다.

기존에 우리가 high level에서만 구현하던 다른 프로그래밍 환경들과 달리, 시스템과 하이레벨 코드의 간의 상호작용을 직접 확인하며 구현해야 했기에 기존에 작성해오던 코드들과 사뭇 다른점들이 상당히 많아서 어려움을 많이 겪었던 것 같습니다.

우선 하이레벨 환경과 같이 사소한 변수들은 자동으로 조정해주는 것이 아닌, 컴퓨터의 관점에서 작은 변화도 크게 허용되는 것을 직접 사용해보면서 체감하여 기존에 프로그래밍을 해오던 방식과 다른 방식에 적응하는 것의 필요성이 느껴졌습니다.

이 과정에서 Syscall.c 와 syscall.h 에 내가 새로 추가한 함수를 등록하여 시스템이 내가 새롭게 추가한 함수를 인지하게 만들 수 있다는 것을 배웠고, user.h와 실행에 필요한 유저코드를 짜면서 어떻게 시스템과 상호작용하는 코드를 작성하는 것인지에 대한 개념을 일부분이나마 알게 되었습니다.

또한 이론에서 간접적으로 배운 것들을 직접 디자인하고 구현해보니 단순히 구현에 성공, 실패한 것을 떠나 많은 것들을 알게 되었습니다. 우선 cscope를 통해 구현에 도움이 될만한 함수와 변수들을 참고하는 과정에서 다양한 헤더와 파일들을 순차적으로 옮겨 다니며 내가 구현한 코드가 어떻게 전달되는지가 명확하게 눈에 들어왔습니다.

이렇게 직접 해보며 개념들을 접하자 교수님의 이론시간에 중요하다고 들었던 interrupt 등 다양한 시스템 콜들의 동작과정들이 어느정도 눈에 보이게 되어 이론을 들을 때 접하는 개념들이 조금 더 명확하게 머릿속에 들어가게 되어 많은 도움이 된 것 같습니다.