**운영체제 2장**

과제 3

2015253039 권진우

**실습연습문제**

2.1 System-call 의 목적은 무엇인가?

-System-call을 통해서 유저모드 레벨의 프로그램이 특권 명령어를 사용할 수 있는 OS에게 서비스를 요청할 수 있다. OS가 서비스를 하도록한다.

2.5 Command interpreter의 목적은 무엇인가? 왜 커널과는 구분을 짓는가?

-command interpreter는 사용자나 명령 파일로부터 실행하고, 보통 하나 또는 그 이상 갯수의 system-call로 전환되어 실행된다. command interpreter는 변경 될 수 있으므로 일반적으로 커널의 일부가 아니다.

2.6 새로운 프로세스를 시작하기 위해 어떤 System-call을 command interpreter나 shell에 의해 실행해야 하는가?

-유닉스 시스템에서는 새로운 프로세스를 시작하기 위해 folk라는 system-call을 호출에 이어 실행 system-call 호출을 수행해야 한다. folk call은 현재 실행중인 프로세스를 복제하는 반면, 실행 call은 호출 프로세스에 대해 다른 실행 파일을 기반으로 새로운 프로세스를 오버레이한다.

2.10 운영체제를 펌웨어(firmware)에 저장하고 다른 프로그램들은 디스크에 저장하는 이유는?

-펌웨어는 잘 수정되기 어렵고 휴대용 PDA 및 휴대폰과 같은 특정 장치의 경우 파일 시스템이 있는 디스크를 장치에 사용할 수 없을 수도 있다. 따라서 운영체제는 펌웨어에 저장되어야 한다.

**2장 연습문제**

2.17 운영체제가 제공하는 시스템 호출 인터페이스를 사용하여 사용자가 새로운 명령어 해석기를 개발하는 것은 가능한가?

-입출력 장치와 파일들 간에는 유사성이 매우 많기 때문에 많은 운영체제가 이들 둘을 통합된 파일-장치 구조로 결합하였다. 이 경우 같은 시스템 호출들이 파일과 장치에 사용된다. 따라서 시스템 호출은 다르더라도 사용자 인터페이스(UI)를 통해서 새로운 명령어 해석기와 같은 인터페이스를 구축하는 것이 가능하다.

2.19 기법과 정책을 분리하는 것이 바람직한 이유는?

-빌딩 블록의 기본 집합을 구현함으로써 기법과 정책 분리를 추구하는데, 이 블록들은 정책으로부터 거의 자유롭고, 보다 고급의 기법과 정책들이 사용자 생성 커널 모듈이나 사용자 프로그램 자체를 통해 첨가될 수 있기 때문이다. 또한 정책과 기법 분리는 융통성을 위해 아주 중요하다. 정책은 장소나 시간에 따라 바뀔 수 있기 때문이다. 기법-어떻게 할것인가, 정책-무엇을 할것인가

2.21 시스템을 설계할 때 마이크로커널 방식을 사용하느 장점은 무엇인가? 마이크로 커널 구조에서 사용자 프로그램과 시스템 서비스가 상호작용하는 방식에 대해 설명하시오. 마이크로커널 방식을 사용할 때의 단점은 무엇인가?

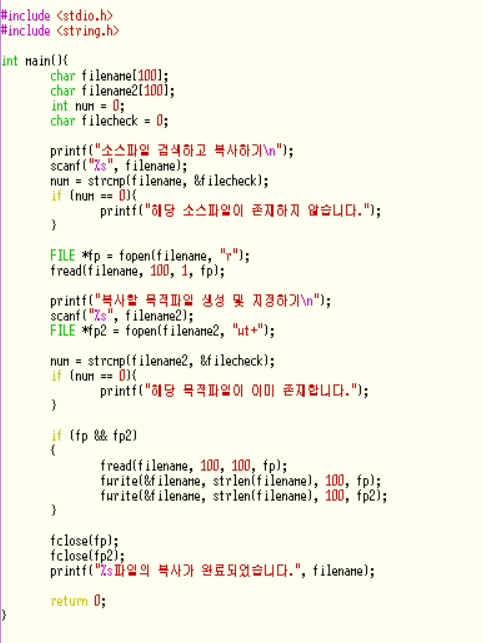
-마이크로 커널은 계층적구조인 Layered 커널(계층별 시스템 호출을 하고 계층적 구조를 가짐으로 인해서 커널의 용량(크기)가 커지고 무거워져서 설계하는데 어려운 면)을 갖고 있는 것과는 달리 이를 해결하기 위해 사용자 모드로 상당히 많은 주요 기능들을 빼낸 커널에 꼭 필요한 부분만 넣은 Microkernel(smallkernel)이 등장하였고 사용자 모듈에서 필요 시 마다 커널로 message-passing기능을 통해서 서비스를 요청하는 방식을 사용한다. 이로써 발생하는 장점은 OS확장이 용이해졌고 이식이 용이하며 높은 신뢰성+보안성을 가진다.(서비스가 잘못되더라도 사용자 프로그램에서 잘못될 확률이 큼) 단점으로는 사용자 모드와 커널 모드간 통신 오버헤드가 발생함으로써 성능이 좋지는 못함.

2.22 적재 가능 커널 모듈을 사용하는 장점은?

적재 가능 커널 모듈(Loadable kernel module)을 사용하면 꼭 필요한 것만 커널에 넣어두고 필요시 마다 선택적으로 실행시간 동안 동적으로 적재되어 커널 기능을 확장하기 쉽고 서로 프로그램,시스템간에 통신 오버헤드가 발생하지 않아서 성능 또한 보장된다. 유연하고 효율적인 장점이 있다.

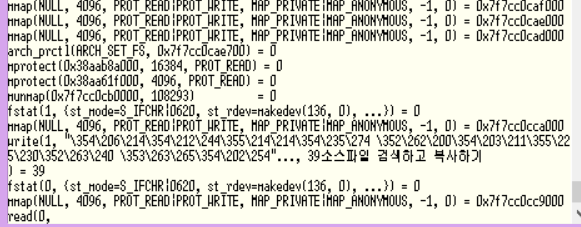
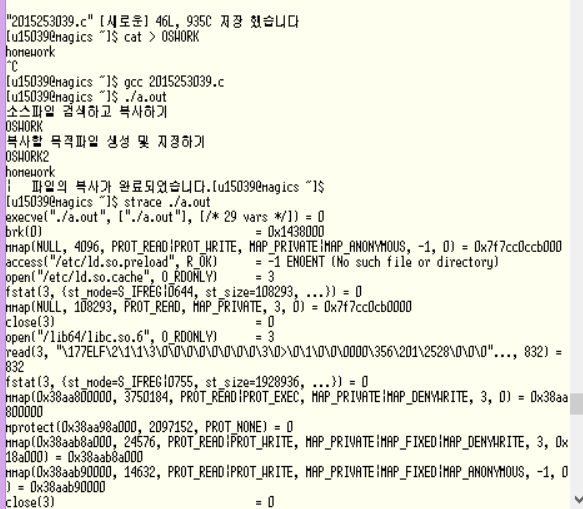
<다음장에 계속 3번 문제>

**3.** 2장 2.26번의 programming problem(파일 복사)을 Linux에서 작성하시오. Linux에서 system-call을 추적하는 유틸리티로 STrace를 사용한다. 그리고 STrace 명령어의 기능도 알아보시오.



<- 리눅스에서 작성한 파일 복사

기능 코드



-리눅스에서 파일 복사 프로그램을 실행 중 STrace ./a.out을 이용하여 System call을 이용하여 실행 중 일어난 System call을 모두 보여주는 기능을 사용해보았다. 또한 Strace에 추가적인 기능으로 -를 사용하면,

-c -- count time, calls, and errors for each syscall and report summary

-f -- follow forks, -ff -- with output into separate files

-F -- attempt to follow vforks, -h -- print help message

-i -- print instruction pointer at time of syscall

-q -- suppress messages about attaching, detaching, etc.

-r -- print relative timestamp, -t -- absolute timestamp, -tt -- with usecs

-T -- print time spent in each syscall, -V -- print version

-v -- verbose mode: print unabbreviated argv, stat, termio[s], etc. args

-x -- print non-ascii strings in hex, -xx -- print all strings in hex

-a column -- alignment COLUMN for printing syscall results (default 40)

-e expr -- a qualifying expression: option=[!]all or option=[!]val1[,val2]...

options: trace, abbrev, verbose, raw, signal, read, or write

-o file -- send trace output to FILE instead of stderr

-O overhead -- set overhead for tracing syscalls to OVERHEAD usecs

-p pid -- trace process with process id PID, may be repeated

-D -- run tracer process as a detached grandchild, not as parent

-s strsize -- limit length of print strings to STRSIZE chars (default 32)

-S sortby -- sort syscall counts by: time, calls, name, nothing (default time)

-u username -- run command as username handling setuid and/or setgid

-E var=val -- put var=val in the environment for command

-E var -- remove var from the environment for command

다음과 같은 기능을 사용할 수 있다.