**운영체제 3장**

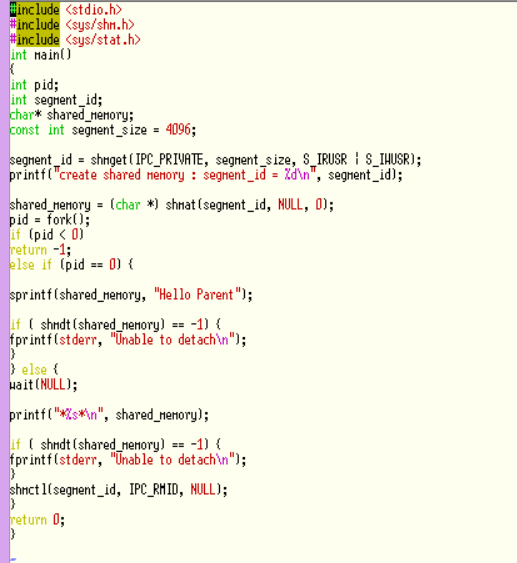
과제5

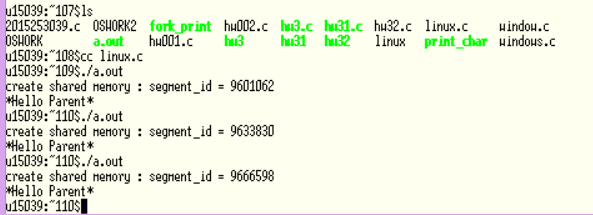
2015253039 권진우

**1. (IPC)**

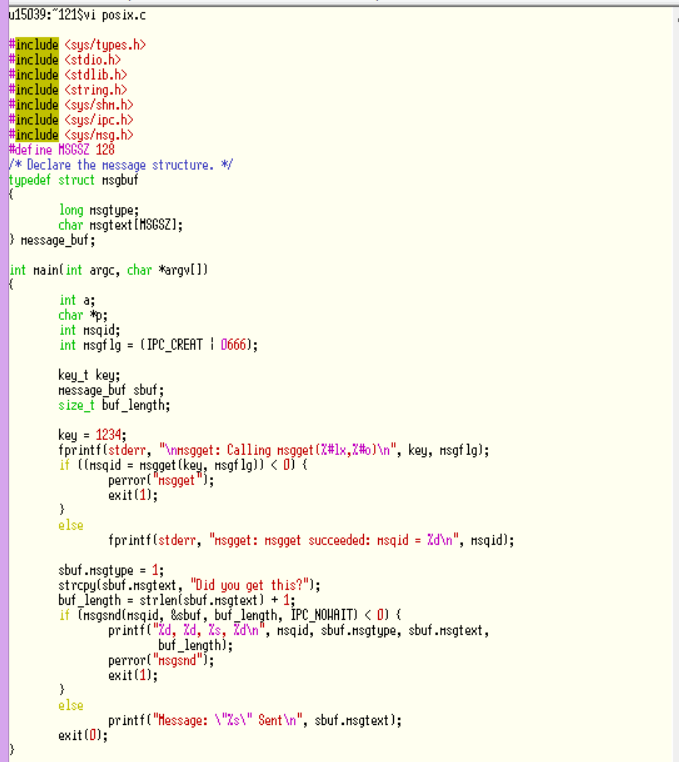
**(1)** 뒤에 첨부한 프로그램을 분석하고 실행시켜보시오.

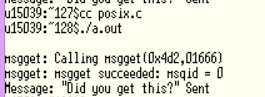
**리눅스 공유메모리 할당 코드**

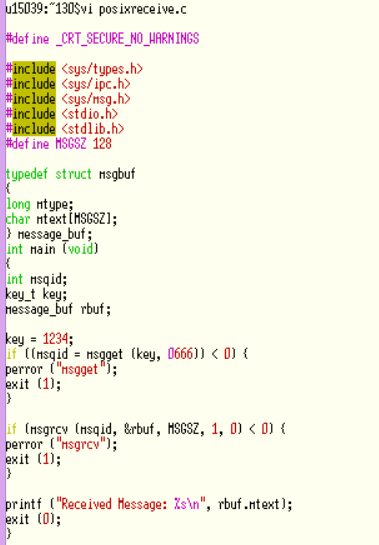


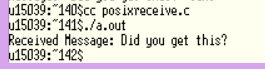
**리눅스 공유 메모리 할당**

-매번 공유 메모리(segment\_id)의 주소가 바뀜

 **posix 메시지 전송 코드**

**posix 메시지 전송 실행**

**posix 메시지 받기 코드**

**posix 메시지 받기 실행.**

**-**리눅스 환경에서 공유 메모리 방식의 경우 공유 메모리를 정해줄 pid의 포인터를 정해주고 공유 메모리 사이즈를 4096바이트로 설정하며 공유 메모리를 통해 프로세스가 접근하고 데이터를 가져간다. 이때 매번 segment\_id의 주소가 바뀌어 접근한다. 메시지 전송 방식의 경우 버퍼를 사용하고 OS의 서비스를 받아서 메시지를 커널에 보내고 다른 필요한 프로세스로 다시 전송한다.

**(2)** 그리고 이 프로그램과 각종 자료들(과목 홈페이지 게시 자료, 인터넷 검색자료 포함)을 참고 하여 POSIX의 shared memory와 message queue를 사용한 interprocess communication 기능 에 대한 API들을 소개하고 두 방법을 비교 설명하시오.

-분석 : **프로세스와 프로세스간 통신은** **message passing** 방식과 **Shared memory** 방식이 있는데 먼저, Message passing 방식은 프로세스에서 System Call(message queue)를 통해서 OS에게 메시지를 전송함으로써 서비스를 받고 OS는 받은 메시지를 다시 메시지를 받을 프로세스로 전달 시킨다. 따라서 적은 크기의 데이터를 Message를 통해 전송하고 받기 편하고 충돌이 없으며 구현이 상대적으로 쉬운 장점이 있다. 코어 수 증가에 따른 속도 증가도 기대할 수 있다. 많은 양의 데이터를 전송하면 복사를 여러 번 많이 해야함으로 인해 비효율성이 증가할 수 있다. 반면, Shared Memory 방식은 메모리에 공유 메모리공간을 만들어 버퍼 역할을 하고 버퍼의 크기를 지정하고 각 프로세스에서 공유메모리로 데이터를 전송하고 받음으로써 message passing방식보다 더욱 빠르고 공유 메모리의 영역을 지정할 때만 OS에 System-call을 보냄으로써 보통메모리처럼 접근이 가능하다. 반면 단점으로는 코어 수 증가 시 캐시 일관성 문제로 성능저하가 일어날 수 있다.

-Send(message) : 메시지 전달(보내기)

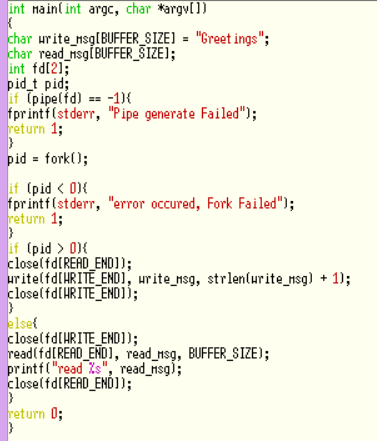
receive(message) : 메시지 받기

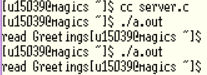
shmdmt() : 공유 메모리 연결 해제

shmget() : 공유 메모리 생성 or 접근, size를 통해 byte 단위로 지정, 커널에서 성공적으로 공유메모리 공간을 할당하게 되면 공유 메모리를 가리키는 식별자를 리턴.

 IPC\_CREAT : shared memory가 없으면 새로 생성

**2.** (pipe) Unix/Linux pipe와 관련하여 교과서 3장의 Figure 3-25 프로그램을 작성하여 실행시키 고, Unix/Linux에서의 pipe 기능의 사용법에 대해서 알아보시오.

pipe를 사용한 프로그램 생성(변수에 띄어쓰기X , 대신에 \_사용)



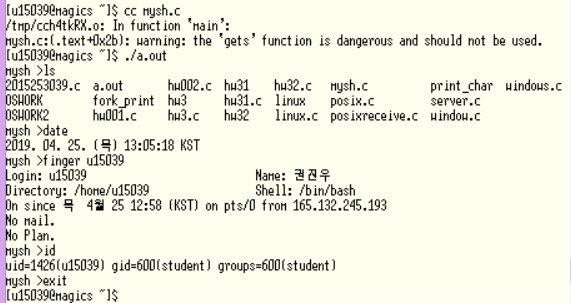
-Pipe는 두개의 프로세스가 서로 통신이 가능하도록 전달자 역할을 수행하며 Linux에서 파이프의 기능은 프로그램 생성에 있어서 부모 프로세스가 자식 프로세스를 생성할 때 먼저, Pipe를 만들어서 데이터 전송 통로를 만들고 자식을 프로세스를 생성하여서 생산자인 부모로부터 소비자인 자식프로세스로 데이터가 Pipe를 통해 Write되고 Read된다. pid가 0보다 클때는 부모 프로세스를 말하여 read\_end파이프를 폐쇄시킨다. 반대로 pid가 0인 경우 자식을 가리키고 write\_end를 폐쇄한다. 일반 Pipe는 단방향 통신만 가능하고 이때 부모로부터 연결되는 fd는 write-end에서만 가능하고 read-end는 폐쇄시킨다. 반대로 자식프로세스로 연결 되는 fd는 read-end에서만 가능하게 만들어진다.

지명 Pipe의 경우, 양방향 통신이 가능하고 부모-자식 관계가 필요가 없으며 여러 프로세스들이 지명 Pipe를 통해서 통신이 가능하다. 파이프 형태의 파일을 만들어서 그 파일을 통해서 데이터를 write하고 read한다.

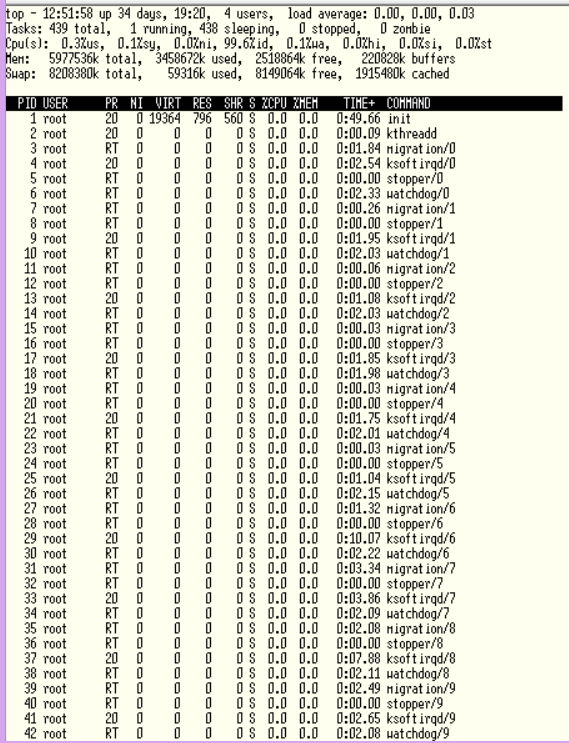
**3.** (간단한 shell 작성) UNIX/Linux 운영체제 환경에서 명령어를 입력받아서 실행시키는 기능을 갖는 프로그램 mysh을 작성하시오. (힌트: mysh에서 fork와 execvp를 사용하여 입력받은 명령 어를 실행시킨다. 명령어의 인수가 가변개수이므로 execlp보다 execvp를 사용하는 것이 편리하 다, exec계열 시스템 호출 사용법은 [process] 참고자료 및 인터넷 자료를 참조하시오.) 이 shell은 exit 명령어 또는 Ctrl+D를 입력받으면 종료한다. (동작 확인을 할 때에 다양한 Linux 명령어를 입력해보시오.)



-execvp를 이용한 명령어 실행 코드



-mysh.c 프로그램을 실행하여 다양한 명령어 실행



-mysh.c 프로그램을 이용한 top 명령어 실행 화면

-명령어

ls : 파일 목록 출력 , id : 사용자 id 출력

date : 현재 날짜 및 시간 출력, finger : 개별 사용자 정보 확인 명령어

top : 현재 프로세스 관련 목록 출력(CPU 및 메모리 사용량)

finger 이외 who, users 등이 있습니다.