1주차 2차시 리눅스 시스템 프로그래밍

【학습목표】

1. 리눅스 시스템 프로그래밍에 대해 설명할 수 있다.
2. 시스템 콜에 대해 설명할 수 있다.

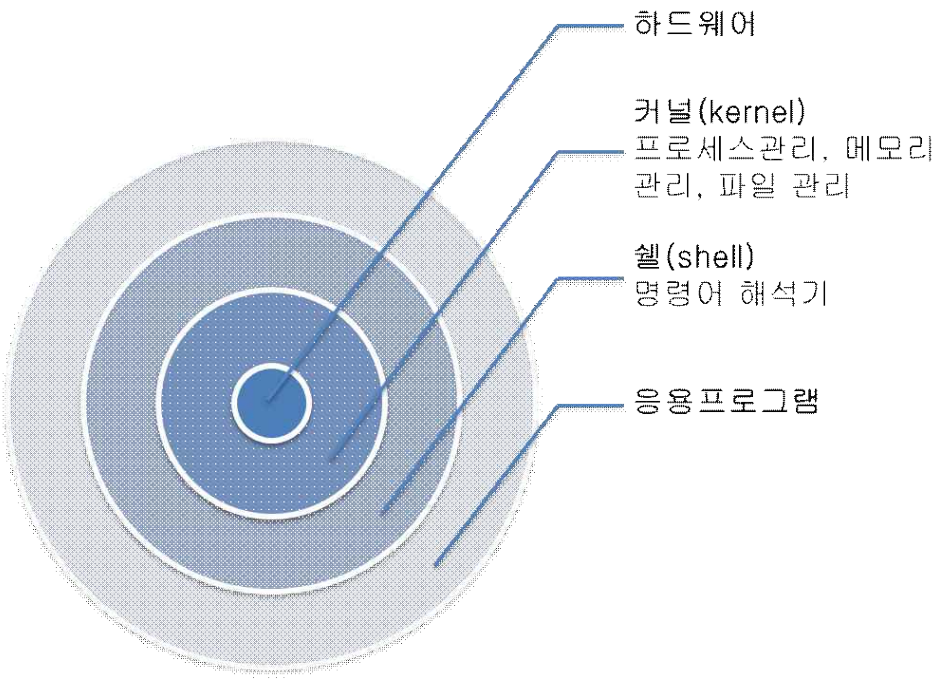
학습내용1 : 리눅스 시스템 프로그래밍의 개요

1. 리눅스의 구조

① 커널 : 리눅스의 핵심
- 프로세스/메모리/파일 시스템/장치 관리
- 컴퓨터의 모든 자원 초기화 및 제어 기능

② 셸 : 사용자 인터페이스
- 명령 해석
- 프로그래밍 기능

③ 리눅스 기본 셸: 배시 셸(리눅스 셸)
- 응용 프로그램
- 각종 프로그래밍 개발 도구
- 문서 편집 도구
- 네트워크 관련 도구 등



학습내용2 : 유닉스/리눅스 시스템 프로그래밍의 정의

*** 유닉스/리눅스에서 제공하는 시스템 호출(시스템 콜, 시스템 함수)을 사용해 프로그램을 작성하는 것을 의미**

1. 시스템 호출

- 유닉스 시스템이 제공하는 서비스를 이용해 프로그램을 작성할 수 있도록 제공되는 프로그래밍 인터페이스
기본적인 형태는 C 언어의 함수 형태로 제공

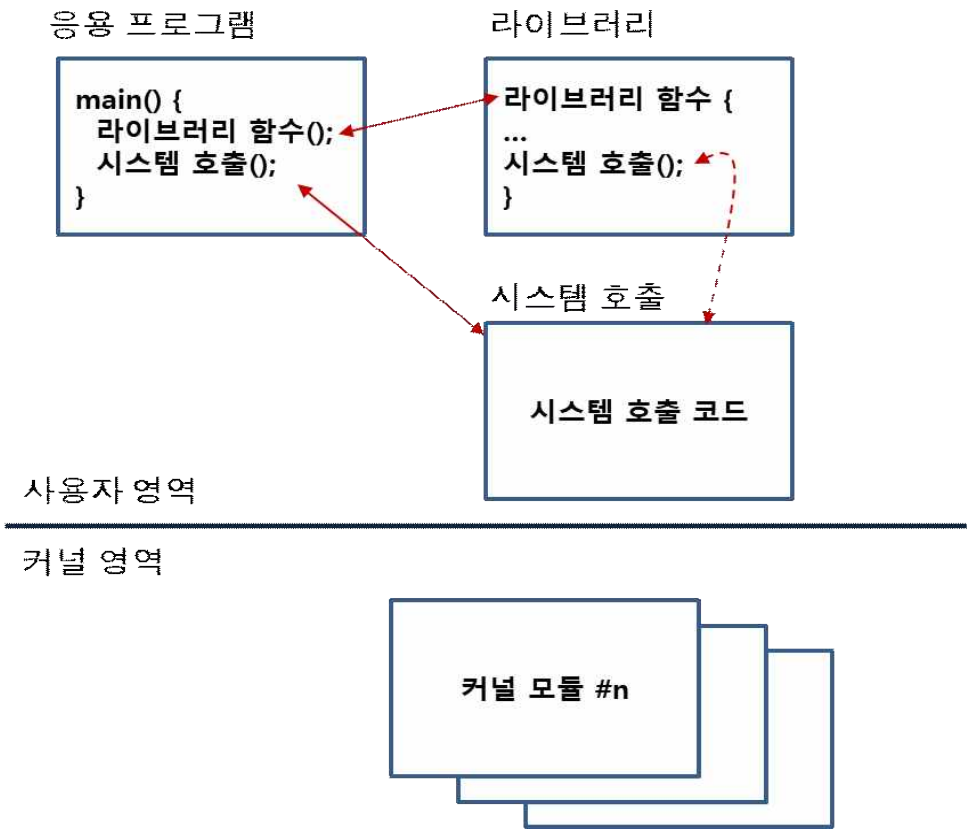
```
리턴값 = 시스템 호출명(인자1, 인자2, .....);
```

2. 라이브러리 함수

- 라이브러리 : 미리 컴파일된 함수들을 묶어서 제공하는 특수한 형태의 파일
- 자주 사용하는 기능을 독립적으로 분리하여 구현해둠으로써 프로그램의 개발과 디버깅을 쉽게하고 컴파일을 좀 더 빠르게 할 수 있다
- /lib, /usr/lib에 위치하며 lib*.a 또는 lib*.so 형태로 제공

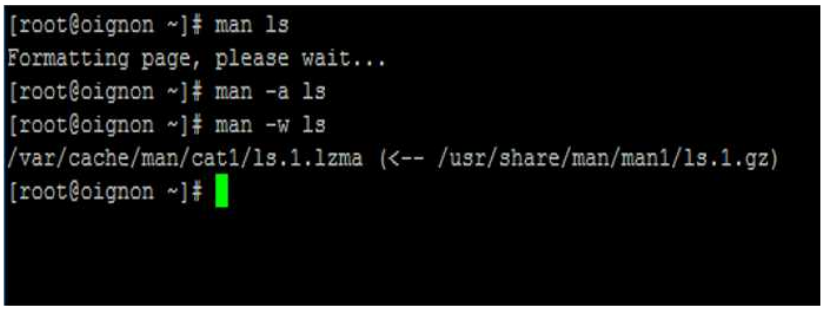
3. 시스템 호출과 라이브러리 함수 비교 1

- 시스템 호출 : 커널의 해당 서비스 모듈을 직접 호출하여 작업하고 결과를 리턴
- 라이브러리 함수 : 일반적으로 커널 모듈을 직접 호출안함



4. man

- 리눅스가 제공하는 각종 명령의 사용법을 보여줌

형식 :	명령어	-옵션	인자1 인자2..
	man		ls
사용예			

```
[root@oignon ~]# man ls
Formatting page, please wait...
[root@oignon ~]# man -a ls
[root@oignon ~]# man -w ls
/var/cache/man/cat1/ls.1.lzma (<-- /usr/share/man/man1/ls.1.gz)
[root@oignon ~]#
```

5. 시스템 호출과 라이브러리 함수 비교 2

- 시스템 호출 : man 페이지가 섹션 2에 속함

```
System Calls                                                    open(2)

NAME
        open, openat - open a file
SYNOPSIS
        #include <sys/types.h>
```

- 라이브러리 함수 : man 페이지가 섹션 3에 속함

```
Standard C Library Functions                                    fopen(3C)

NAME
        fopen - open a stream
SYNOPSIS
        #include <stdio.h>
```

6. 시스템 호출과 라이브러리 함수 비교 3

* 시스템 호출의 오류 처리방법
- 성공하면 0을 리턴, 실패하면 -1을 리턴
- 전역변수 errno에 오류 코드 저장 : man 페이지에서 코드값 확인 가능

[예제 1-1] 시스템 호출 오류 처리하기 ex1_1.c

```
01  #include <unistd.h>
02  #include <stdio.h>
03
04  extern int errno;
05
06  int main(void) {
07      if (access("unix.txt", F_OK) == -1) {
08          printf("errno=%d\n", errno);
09      }
10
11      return 0;
12  }
```

```
# ex1_1.out
errno=2
```

```
# vi /usr/include/sys/errno.h
......
/*
* Error codes
*/
#define EPERM   1       /* Not super-user */
#define ENOENT  2       /* No such file or directory */
......
```

6. 라이브러리 함수의 오류 처리방법

- 오류가 발생하면 NULL을 리턴, 함수의 리턴값이 int 형이면 -1 리턴
- errno 변수에 오류 코드 저장

[예제 1-2] 라이브러리 함수 오류 처리하기

```
01  #include <stdlib.h>
02  #include <stdio.h>
03
04  extern int errno;
05
06  int main(void) {
07      FILE *fp;
08
09      if ((fp = fopen("unix.txt", "r")) == NULL) {
10          printf("errno=%d\n", errno);
11          exit(1);
12      }
13      fclose(fp);
14
15      return 0;
16  }
```

```
# ex1_2.out
errno=2
```

man fopen에서 확인

학습내용3 : C언어 개요

1. C언어란?

- 1972년, 데니스 리치
- 뛰어난 기능과 융통성을 제공
- 대부분의 시스템 소프트웨어를 구현하는 언어
- 고급 언어면서 저급 언어처럼 비트나 바이트 처리, 그리고 포인터에 의한 주소 처리

2. C 언어를 알아야 하는 이유

- 리눅스의 대부분이 C언어로 작성
- 리눅스 환경의 대부분의 애플리케이션을 C언어로 작성
- 결국 리눅스 환경에서 프로그래밍을 공부하거나 개발하려는 분들은 리눅스와 영원히 함께 할 C언어를 이용하는 것이 바람직하다.

3. 리눅스 시스템 프로그래밍의 C언어 관련 도움사이트

사이트명	설명
http://gcc.gnu.org	gcc 매뉴얼과 프로그램 제공
http://www.gnu.org/software/make	make 매뉴얼과 프로그램 제공
http://www.gnu.org/software/gdb	gdb 매뉴얼과 프로그램 제공
http://www.tldp.org	리눅스와 관련된 기술 자료들을 제공
http://kldp.org	리눅스와 관련된 기술 자료들을 제공(한글)
http://lug.or.kr	리눅스 팁을 제공, Q&A 코너 운영

【학습정리】

1. 유닉스/리눅스 시스템 프로그래밍이란 유닉스/리눅스에서 제공하는 시스템 호출(시스템 콜, 시스템 함수)을 사용해 프로그램을 작성하는 것을 의미한다.

2. 호출의 오류 처리방법은 성공하면 0을 리턴, 실패하면 -1을 리턴한다.

3. 라이브러리 함수의 오류 처리방법은 오류가 발생하면 NULL을 리턴하고 함수의 리턴값이 int 형이면 -1 리턴한다.

4. 리눅스의 대부분이 C언어로 작성되어 있다.