UNIT

04

리눅스 시스템 개요

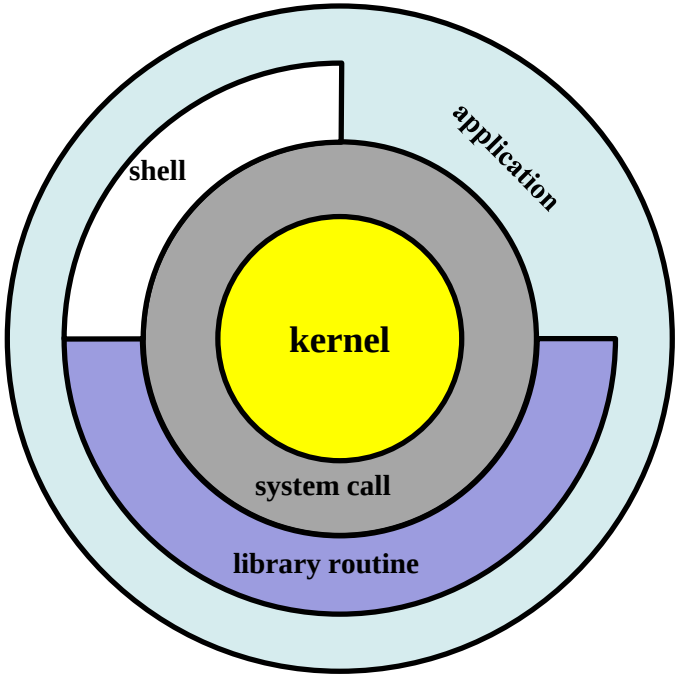
로봇SW 교육원

최상훈(shchoi82@gmail.com)

학습 목표

- 리눅스의 구조 이해
- 리눅스 파일 I/O와 프로세스의 개념 이해

Linux 구조



- OS(Operating System)
  - **컴퓨터 시스템 자원관리자**
  - **컴퓨터 자원을 추상화 시키는** S/W
  - **컴퓨터 하드웨어 자원을 제어**
  - S/W **실행될 수 있는 환경 제공**
- Kernel
  - **운영환경의 핵심** S/W
- System call
  - **커널에 접근하기위한 인터페이스**, **진입점**(entry point)
- Shell
  - **사용자의 명령을 분석하고 실행하는 명령어 해석기**(command interpreter)
- Library
  - **응용**S/W **에서 공통적으로 쓰이는 함수들**

Logging

- Linux **접속 시 사용자 계정 확인(로그인 프로세스)**
- **사용자계정 확인(사용자ID와 패스워드)**
- **로그인 프로세스는 사용자 정보를 파일에 저장함**
  - /etc/passwd**에 계정정보를 저장함**
    - **계정 정보가 ':' 을 구분으로 저장되어 있음**

name:password:uid:gid:comment:homepath:shellc

```
20 pi:x:1000:1000:,,,:/home/pi:/bin/bash
```

  - /etc/shadow
    - **암호화된 패스워드 저장**
  - /etc/group
    - **그룹 정보 저장**

파일

- 리눅스 파일 시스템
  - 다양한 파일들이 디렉토리를 통해 계층적으로 구성되어 관리됨
  - 트리구조
    - 재일 상위 디렉토리 : 루트 디렉토리
  - 대표적인 리눅스 파일 시스템
    - ext3, ext4

- 파일
  - 리눅스는 모든 자원을 파일로 취급함
  - 다양한 파일의 종류가 있음
    - 일반 파일, 디렉토리, 장치 파일, 소켓, 파이프, 심볼릭 링크 파일
  - 디렉토리
    - 다른 파일들과 디렉토리의 목록이 저장되어 있는 일종의 파일

- 파일경로
  - 상대경로와 절대경로
  - 현재 경로
    - 상대경로의 기준이 됨

디렉토리

- **현재 (작업)디렉토리**(Current Working Directory) **경로**
  - **상대경로의 기준이 되는 경로**
  - **각 프로세스의 저장된 속성**
  - pwd**명령으로 확인**
- **현재 (작업)디렉토리 병경** : cd, chdir **명령**
- **디렉토리 생성** : mkdir **명령**
  - **자동적으로 생성되는 디렉토리 항목**
    - . : **현재 디렉토리**
    - .. : **부모 디렉토리**
- **홈 디렉토리** : **로그인 했을 때 시작** CWD **경로**
  - /etc/passwd **파일에 명시됨**
  - echo $HOME

실습1:파일과 디렉토리

- myls.c : **명령줄로 전달된 디렉토리 경로의 파일 목록 출력**

```
#include<stdio.h>
#include<dirent.h>

int main(int argc, char *argv[])
{
    DIR *dp;
    struct dirent *dirp;

    if (argc != 2){
        printf("usage: ./a.out <directory_name>\n");
        return 1;
    }
    if ((dp = opendir(argv[1])) == NULL){
        printf("can't open %s", argv[1]);
        return 1;
    }
    while ((dirp = readdir(dp)) != NULL)
        printf("%s\n", dirp->d_name);

    closedir(dp);
    return 0;
}
```

파일 I/O

- **파일 디스크립터** (file descriptor)
  - **프로세스가 파일을 참조하기 위해서 사용하는 식별자**
  - **자동으로 열리는 파일 디스크립터들**
    - **표준 입력 / 표준 출력 / 표준 에러**

- Unbuffered I/O **와** Buffered I/O(Standard I/O)

실습2:파일 I/O

- Standard **파일** I/O

```
#include<stdio.h>
int main(void)
{
    int c;
    while((c = getc(stdin)) != EOF)
        if(putc(c, stdout) == EOF){
            printf("output error\n");
            return 1;
        }
    if(ferror(stdin)){
        printf("input error\n");
        return 1;
    }
    return 0;
}
```

```
$ ./a.out > data
hello world[Ctrl+d][Ctrl+d]
$ cat data
hello world
$ ./a.out < data > data.copy
$ cat data.copy
hello world
$
```

실습3:파일 I/O

- unbuffered **저수준 파일** I/O

```
#include<stdio.h>
#include<unistd.h>
#define BUFFSIZE    4096

int main(void)
{
    int n;
    char    buf[BUFFSIZE];

    while ((n = read(STDIN_FILENO, buf, BUFFSIZE)) > 0)
        if (write(STDOUT_FILENO, buf, n) != n){
            printf("write error\n");
            return 1;
        }

    if (n < 0){
        printf("read error\n");
        return 1;
    }
    return 0;
}
```

```
$ ./a.out > data2
hello world[Ctrl+d][Ctrl+d]
$ cat data2
hello world
$ ./a.out < data2 > data2.copy
$ cat data2
hello world
$
```

프로세스

- **프로세스 : 프로그램 하나의 실행 인스턴스**
  ※ **실행되고 있는 프로그램을 태스크(task)라고 부르는 OS도 있음**

- **프로세스 ID : 리눅스 시스템은 모든 프로세스에게 각자 고유한 식별자를 부여하여 프로세스를 관리함**

```
#include<stdio.h>
#include<unistd.h>
int main(void)
{
    printf("Process ID %d\n", getpid());
    return 1;
}
```

```
$ ./a.out
Process ID 17676
$ ./a.out
Process ID 17677
$ ./a.out
Process ID 17678
```

Error Handling

- Error handling
  - **오류발생시 음수 또는** NULL**을 리턴**
  - errno **변수**(extern) : **오류에 대한 추가 정보 제공**
    - Ex) open **함수의 경우** 15**가지의** errno **가지고 있음**
      - **파일이 존재하지 않거나 접근 허가 문제 등**
  - **발생할 수 있는 오류의 종류**
    - man 3 errno

- **오류 관련 함수들**
  - char *strerror(int errnum);
  - void perror(const char* msg);

리눅스 시간 값

- Linux는 두 종류의 시간 값을 사용함

- 달력시간 : 1970년 1월 1일 00:00:00을 기준으로부터의 시간(초단위)
  - 파일의 수정된 시간 등을 설정할 때 사용됨
  - time_t 변수를 사용

- 프로세스 시간: 프로세스의 CPU 사용 시간정보
  - CPU의 clock을 사용
  - clock_t 변수를 사용
  - 프로세스관련 세가지 시간 값
    - Clock 시간(wall clock time)
      - 프로세스가 실행된 시간
    - 사용자 CPU 시간
      - 사용자 수준 명령들을 수행에 소비한 CPU시간
    - 시스템 CPU 시간
      - 프로세스의 요청에 의해 커널이 소비한 CPU 시간

시스템 콜과 라이브러리 함수

- **시스템 콜**
  - **잘 정의되어진** kernel**로의 진입점**(entry point)
  - man page 2 **섹션**
  - **일반** C **함수처럼 보이기**
    때문에 단순히 헤더파일만 포함하여 호출하면 됨

- **라이브러리 함수**
  - **공통적으로 사용하는 함수들**
  - **더 정교한 기능을 수행하기**
    위해 사용
  - **내부적으로 시스템 콜을 호출하기도 함**