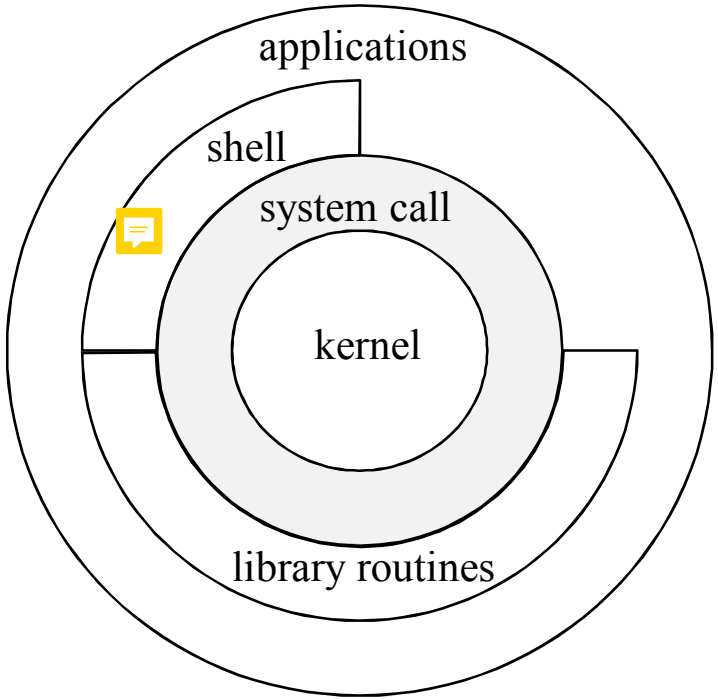
Unix System Overview

Unix architecture

- Architecture of UNIX operating system



Logging in

- Login
  - UNIX는 multi-user system
  - 따라서, “login name + password”로 시스템 로그인
  - /etc/passwd에 다음과 같은 정보 저장.
    - login name:password:UID:GID:comment:home directory:shell

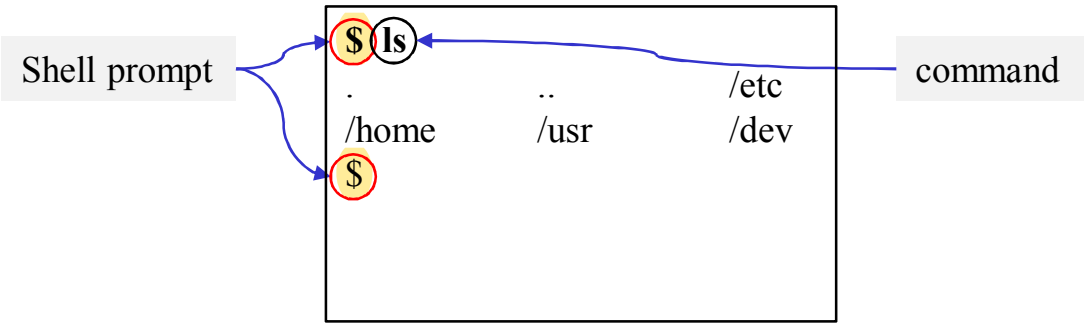
root

```
login as: obama
obama@winter.kw.ac.kr's password:
```

Shell

Shell

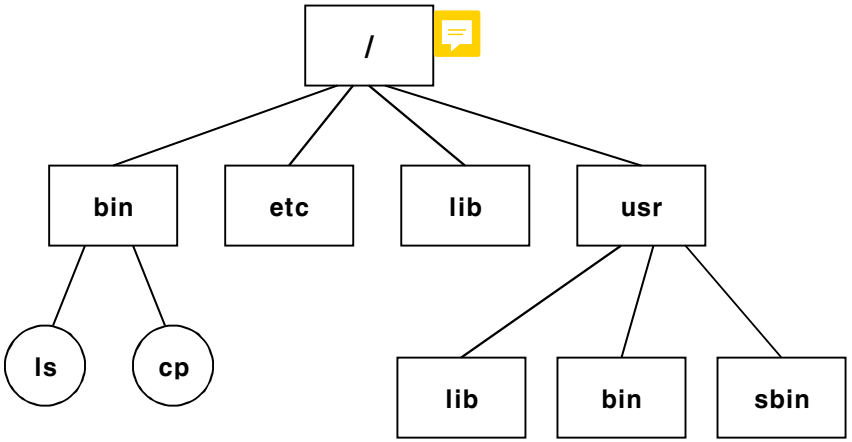
- UNIX는 기본적으로 command line interface를 사용.
- 사용자의 명령을 읽어들여 실행하는 명령어 해석기가 필요.



- Shell의 종류
  - Bourne shell, C shell, Korn shell, Tenex shell, Born Again shell

File system

- File system의 구성
  - File
  - Directory
    - Directory entry(file name + attribute pointer)들의 집합



Hierarchical structure

File system

- Attribute
  - Per file data structure
  - Type, size, owner, permission, access time 등

```
$ ls -al
drwxr-xr-x  13 obama users   4096 Apr  5 12:39 .
drwxr-xr-x  32 obama users   4096 Nov 15 23:38 ..
-rw-r--r--   1 obama users     20 Oct 13  2004 Readme
drwxr-xr-x   2 obama users   4096 Oct 13  2004 adir
$
```

File system

- Filename
  - NULL과 '/'를 제외한 문자열로 구성.
  - BSD의 경우 255자 이내
  - Special filenames
    - . (current directory)
    - .. (parent directory)

File system

Pathname

- /home/obama ➔ home directory (login시 위치)
- /home/obama/test ➔ working directory (작업 위치)

```
$ pwd
/home
$ cd obama
$ pwd
/home/obama
$ cd /usr/bin
$ pwd
/usr/bin
$ cd
$ pwd
/home/obama
$
```

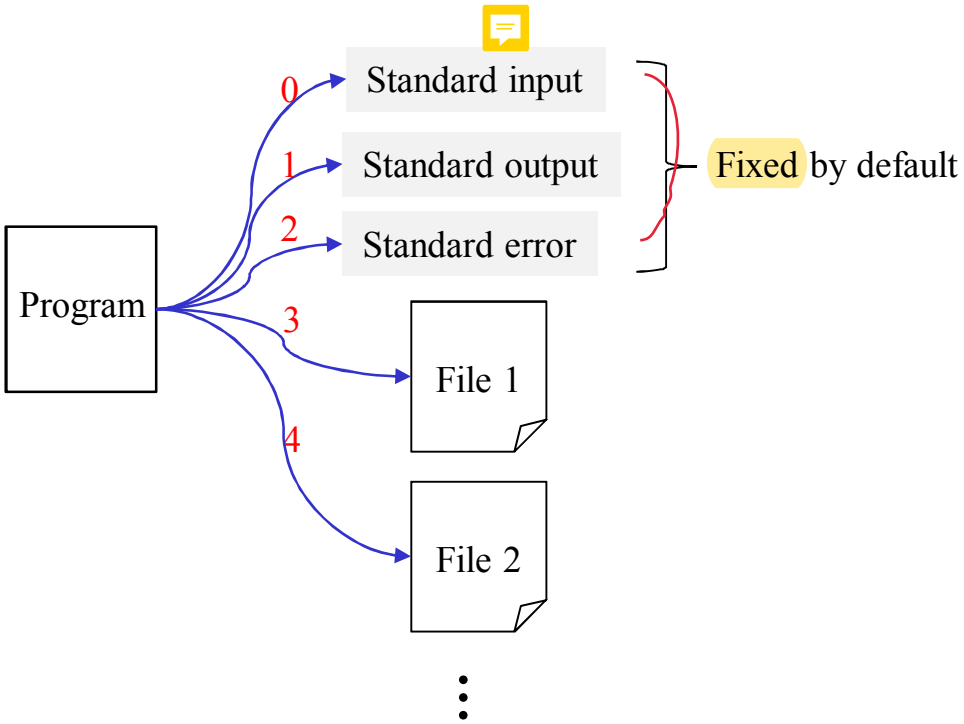
relative path

Absolute path

```
% cd ~          // is equivalent to 'cd'
% cd -          // is equivalent to $OLDPWD
```

Input and output

- File descriptor
  - Non-negative integers that the kernel uses to identify the files being accessed by a process



Programs and processes

- Program
  - An executable file residing on disk
- Process
  - An executing instance of a program (also called task)
  - Process id: a unique numeric identifier for process

```
% ps
  PID  TTY        TIME CMD
24098 pts/2    00:00:00 bash
24162 pts/2    00:00:00 ps
```

Error handling

- When an error occurs,
  - A negative value is often returned
  - errno is set to a value that gives additional information
    - E.g. When open, returns -1 if an error occurs
    - error from open has 15 possible errno values
      - file doesn't exit, permission problem, …
  - <errno.h>
    - Defines errno symbols and constants for each error.
    - Each constant begins with the character E.
      - E.g. ENOENT, EACCESS, …

Error handling

```
#include <string.h>
char *strerror(int errnum);
/* maps errnum(errno value) into an error message string & returns a pointer to the string */
```

```
#include <stdio.h>
void perror(const char *msg);          /* outputs the string pointed to by msg */
                                       /* output format "string pointed by msg: error message" */
```

strerror와 perror의 사용 예

```
#include "apue.h"
#include <errno.h>

int main(int argc, char *argv[])
{
        fprintf(stderr, "EACCES: %s\n", strerror(EACCES));
        errno = ENOENT;
        perror(argv[0]);
        exit(0);
}
```

참고
"apue.h" is widely used in APUE textbook.
- It includes some standard system headers, constants, and function prototypes.

실행 예

```
$ ./a.out
EACCES: Permission denied
./a.out: No such file or directory
```

User identification

- user ID
  - a numeric value that identifies the user
  - is assigned by the system administrator.
  - 0: superuser or root
- group ID
  - is used to collect users together into projects.

- If the full ASCII user/group name is used instead?
  - additional disk space will be required.
  - the cost of string comparison for permission check is high.

User identification

userID와 groupID의 출력 예제

userID와 groupID의 사용 예

```
#include "apue.h"

int main(void)
{
    printf("uid = %d, gid = %d\n", getuid(), getgid());
    exit(0);
}
```

실행 예

```
$ ./a.out
  uid = 205, gid = 105
```

Signals

- Signal
  - is used to notify a process that some condition has occurred.
  - e.g. if divide by zero, SIGFPE(floating point exception) is sent to the process.

- Action of process received the signal
  - ignore the signal.
  - let the default action occur.
  - execute your own action.

Signals

- Signal의 예제
  - kill

```
$ ps
  PID TTY STAT  TIME COMMAND
28478 pp1 S     0:00 -bash
28616 pp1 R     0:02 yes
28617 pp1 R     0:00 ps
$ kill 28616
$ ps
  PID TTY STAT  TIME COMMAND
28478 pp1 S     0:00 -bash
28624 pp1 R     0:00 ps
[1]  + Terminated            yes > /dev/null
$
```

Time values

- calendar time
  - the number of seconds since the Epoch
    - Epoch is 00:00:00 January 1, 1970.
  - time_t

- process time (CPU time)
  - CPU time used by a process (measured in clock ticks.)
    - clock ticks are 50, 60, 100 ticks/seconds.
  - clock_t

Time values

- Process time의 표현
  - clock time
    - the amount of time the process takes to run
    - depends on the number of other processes being run on the system
  - user CPU time
    - CPU time attributed to user instruction
  - system CPU time
    - CPU time attributed to the kernel

```
$ cd /usr/include
$ time -p grep _POSIX_SOURCE */*.h > /dev/null
real    0m0.81s
user    0m0.11s
sys     0m0.07s
```