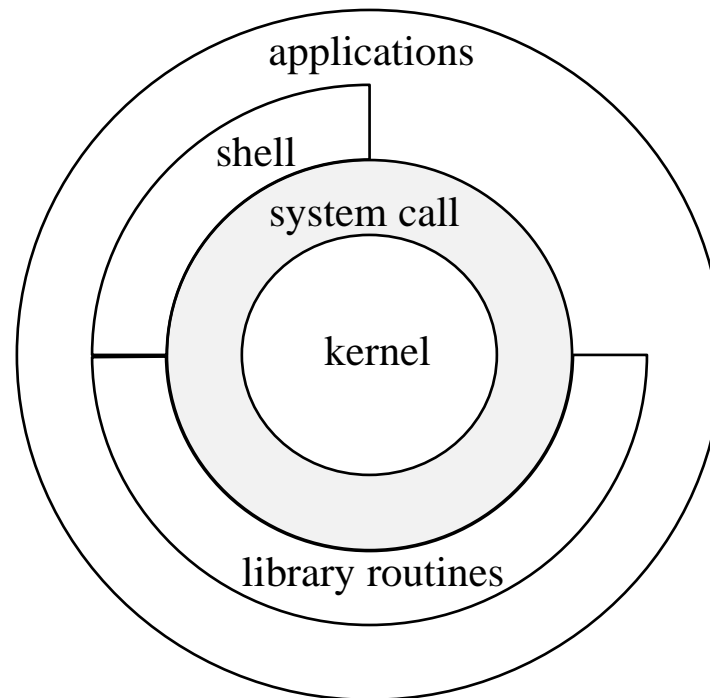

Unix System Overview

Unix architecture

Architecture of UNIX operating system



Kernel : SW to control the HW resources &
provide a program execution environment

Logging in

Login

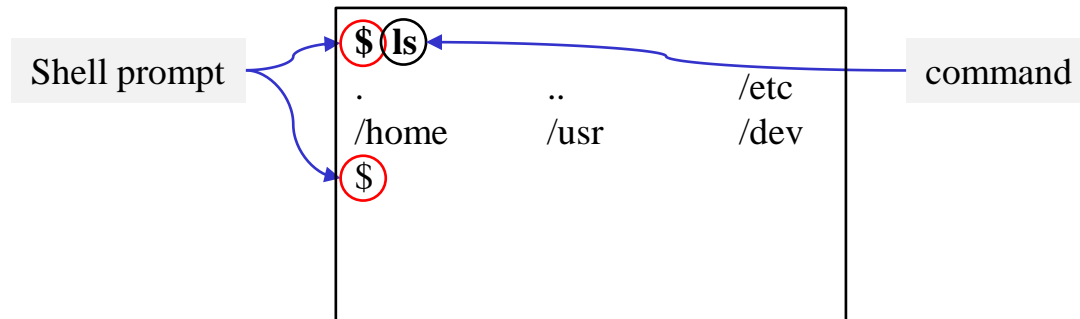
- UNIX is a multi-user system
- Thus, log in to a UNIX system with “login name + password”
- Search the login name in /etc/passwd file composed with
login_name:password:UID:GID:comment:home_directory:shell

```
login as: kwon  
kwon@sp.ulsan.ac.kr's password:
```

Shell

Shell

- A command-line interpreter that reads user input & executes commands

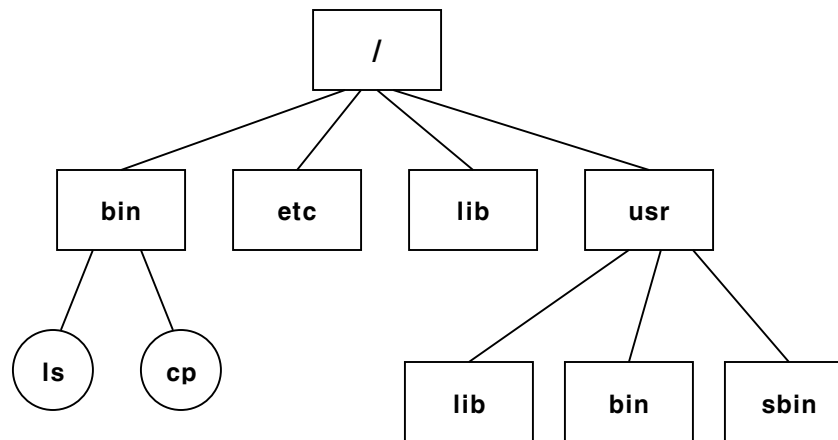


- Shell types
 - Bourne shell, C shell, Korn shell, Tenex C shell, Born Again shell

File system

UNIX File system

- File
- Directory
 - Set of directory entries(file name + attribute pointer)



Hierarchical structure

File system

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

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

File system

Filename

- Excluding the NULL character and ‘/’
- BSD restricts the filename to 255 chars.
- Special filenames
 - . (current directory)
 - .. (parent directory)

File system

Pathname

- Absolute pathname vs. Relative pathname
- /home/obama → home directory (at login time)
- /home/obama/test → working directory

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

relative path

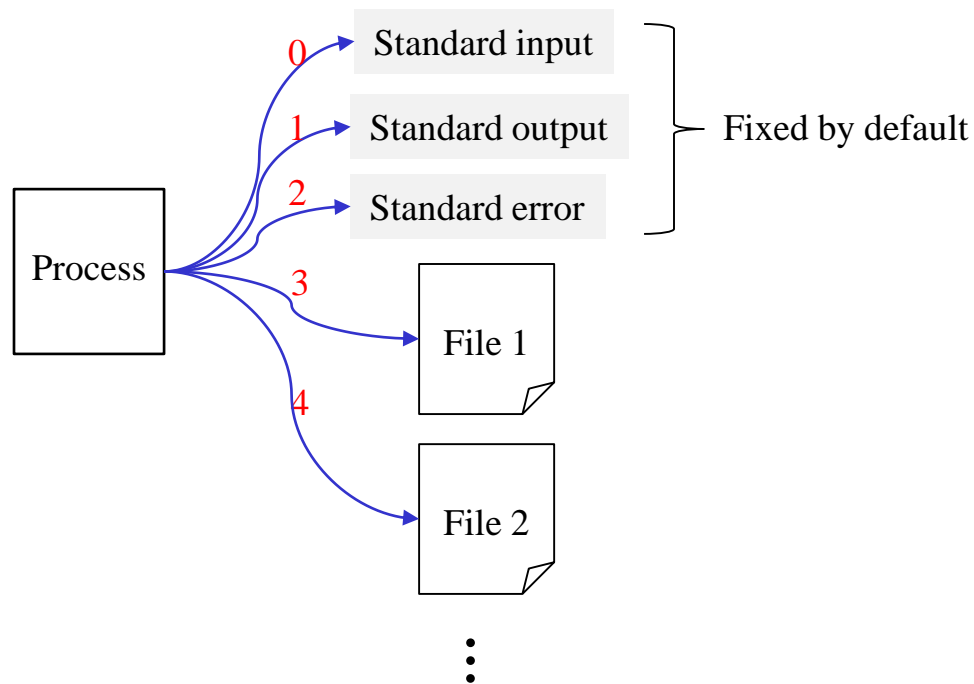
Absolute path

```
% 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. `open()` returns -1 if an error occurs
 - error from `open()` has 15 possible `errno` values
 - file doesn't exist, 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" */
```

Example) strerror & perror

```
#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);
}
```

FYI)

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

```
$ ./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

Example) Print userID & groupID

```
#include "apue.h"

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

Running result)

```
$ ./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

Signal Ex.

- 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

Representation of Process time

- 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
```