CENG328
Operating Systems

Laboratory XII
File System Simulator
&
Review

1. MOSS File System Simulator

- Study the user guide for "File System Simulator".
- Follow the steps below for installation of the software:
  - Create a directory in which you wish to install the simulator (e.g., "moss/filesys").

    mkdir moss
    cd moss
    mkdir filesys
    cd filesys
  - Download the compressed tar archive (filesys.tgz) into the directory.
  - Expand the compressed tar archive.

    tar -zxvf filesys.tgz
  - Export the classpath

    export CLASSPATH=.

1. MOSS File System Simulator

- Using File System Simulator;
  - java mkfs file-name block-size blocks
  - java mkdir directory-path
  - java ls path-name ...
  - java tee file-path
  - java cp input-file-name output-file-name
  - java cat file-name
  - java dump file-name
- See the user guide.
- **Hint:** If you want to compile java codes as given in the manual files, you may be faced with some errors. A suggested solution is that: If there is any import statement that is used for including user-defined class, erase it. (i.e. //import Common;)

1. MOSS File System Simulator

- Use **mkfs** to create a file system with a block size of 64 bytes and having a total of 8 blocks.
  - How many index nodes will fit in a block?
  - How many directory entries will fit in a block?
  - Use **dump** to examine the file system backing file, and note the value in byte 64. What does this value represent?
  - Use **mkdir** to create a directory (e.g., /usr), and then use **dump** to examine byte 64 again. What do you notice?
  - Repeat the process of creating a directory (e.g., /bin, /lib, /var, /etc, /home, /mnt, etc.) and examining with dump.
  - How many directories can you create before you fill up the file system? Explain why.
- Enhance ls.java to display for each file the uid and gid as decimal numbers, and the 9 low-order bits of mode as a 3-digit octal number (i.e., 000..777).

2.1 Review - Signal Handling

- **signal** function,
  - The code below prints out the value of an increasing counter twice a second. Study it.

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

int main()
{
    int count=0;
    while(1)
    {
        printf("%d... ", count++);
        fflush(stdout);
        usleep(500000);
    }

    return 0;
}
```

  - Modify it so that when an interrupt signal is detected, it should toggle between increasing and decreasing the count variable.

2.2 Review - File Access

- Chmod and stat functions,
  - Create a file using:

    touch foo

  - Create a C source file containing the following sample lines:

```
// Create a stat structure for file information
struct stat buf;

// Give read right for owner
chmod("foo", S_IRUSR);

// Print current file access rights
stat("foo", &buf);
printf("foo - %4.4o\n", buf.st_mode & 07777);
```

  - Using the code segment above, write a program that sets access rights of file **foo** to the following. Your program must display results of each modification to the file.

```
r-- r-- r--
r-x r-x r-x
rwx rwx rwx
rwx r-- ---
rw- rw- ---
rw- --x --x
```

2.3 Review - Reading Directories

- **opendir**, **readdir**, **stat** functions,
  - Complete the code below. When executed, it should list all files which are less than 4KiB.
  - The program should accept directory name using command line parameters.

```
DIR *dir;
struct dirent *entity;
struct stat buf;

// Check if a parameter is passed
if (argc != 2) {
    fprintf(stderr, "Usage: %s <directory name>\n", argv[0]);
    exit(EXIT_FAILURE);
}

// Open and check if parameter is a valid directory name
if ((..... = .....) == NULL) {
    fprintf(stderr, "Can not open %s or it is not a directory, terminating.\n");
    exit(EXIT_FAILURE);
}

// Read from directory
while ((..... = .....) != NULL) {
    // Get file information
    stat(....., .....);

    // Check if it is a regular file and if its size is less than 4 KiB
    if (..... && (..... < 4096))
    printf("Name: %s, size=%d\n", ....., (int) .....);
}
```