Unix File System

Files & media	03_Unix_File_System.pdf
Name	Demo 3
Review	☐

What is a file in Unix?

- container of some info
- how to handle devices: system call - performing control & i/o device

How many types of files?

- regular files: an ordinary data file on disk
- special files: a file representing a device, in /dev directory
    - Block special file: devices transferring info in blocks or chunks, just like disks, CD ROM
    - Character special file: devices transferring info in stream of bytes that must be accessed sequentially, e.g., keyboard, printer.
    - FIFO special file: used for inter-process communications (e.g. pipe).
- directories: provide to allow names of files to be

Difference between regular and directory file

- Contents: data vs file info
- Operations: what can be done and who can do them

How are the files in a file system structured?

- hierarchical file
- current working directory

Task 2: current working directory

```
// task2.c
if (getcwd(cwd, sizeof(cwd)) != NULL)
    printf("Current working dir: %s\n", cwd);
else perror("getcwd() error");

#include <unistd.h>
#include <stdio.h>
#include <errno.h>
int main(){
    char cwd[1024];
    if (getcwd(cwd, sizeof(cwd)) != NULL)
        printf("Current working dir: %s\n", cwd);
    else perror("getcwd() error");
    return 0;
}

// operation: gcc -o task2 task2.c
// operation: ./task2

// Result:
// Current working dir: /home/21094655d/test1/lec3
```

How is a file represented in memory and disk?

- i-node / i-number structure
- <name, i-node> == link
  - symbolic link - cross file system
  - hard link - same file system
  - i-node store in hard disk
  - name resolution: pathname → i-node no.
- how large?*
  - 12 direct pointer can point to 12 × 8KB = 96KB of file content.
    A single indirect pointer points to a block of direct pointers. A block can

contain 8KB/4bytes = 2K pointers = 2048
pointers. 2048 direct pointers can point to 2048 × 8KB = 16MB of file content.
A double indirect pointer points to 2048 single indirect pointers, that is 2048 × 16MB = 32GB of file content.
Similarly, a triple indirect pointer points to 64TB of file content.
So one I-node can at the most point to 64TB + 32GB + 16MB + 96KB of file content.

How to find the file using its name?

How to access files from a Unix program?

- how to access
  - file descriptors (in unistd.h)

```
open
read
write
close
fsync
ioctl
```

  - file pointers

```
fopen
fscanf
fprintf
feof
fflush
fread
fwrite
fclose
```

  - Three files are opened automatically:
    STDIN_FILENO: standard input

STDOUT_FILENO: standard output
STDERR_FILENO: standard error

- Accessing files for I/O in three-step process
  - Open the file for I/O
  - Read and write to the file
  - Close the file when finished with I/O

Task 3: Open & read a file

```
int fd = open("someFile", O_RDONLY); // open file
bytes = read(fd, buffer, count(4*buffer_size))


#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
void main(void){
        int fd = open("tmp", O_RDONLY);
        char buf[4];
        int numOfBytesRead = read(fd, buf, 4*sizeof(char));
        int i = 0;
        for (i = 0; i < numOfBytesRead; i++){
                printf("%c", buf[i]);
        }
}
```

Task 4: write a file

```
bytes = write(fd, buffer, count)

#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>

void main(void){
        int fd = open("tmp", O_WRONLY|O_CREAT);
        char* buf = "Hello World!";
        int numBytesWritten = write(fd, buf,strlen(buf));
        printf("%d bytes written to tmp\n", numBytesWritten);
        close(fd);
}

// Operation
// gcc -o task4 task4.c
// ./task4

// result
// 12 bytes written to tmp
```

File pointer

Task 5: fopen()

```
#include <stdio.h>

void main(void){
        FILE * myfp;
        if ((myfp = fopen("tmp","w")) == NULL)
                fprintf(stderr, "Could not fopen file\n");
        else
                fprintf(stdout, "This is a test.\n");
}
// operation
// gcc -o task5 task5.c
```

```
// ./task5

// result
// This is a test.
```

code for file pointer

```
fopen(path, mode)
mode:
r : open file for reading
r+ : open file for reading and writing
w : overwrite file or create file for writing
w+ : open for reading and writing; overwrites file
a : open file for appending (writing at end of file)
a+ : open file for appending and reading

printf(string)
% : escaping variable type
%d,%i : decimal integer
%ld : long interger
%u : unsigned decimal integer
%o : unsigned octal integer
%x,%X : unsigned hexadecimal integer
%c : character
%s : string or character array
%f : float
%e,%E : double (scientific notation)
%g,%G : double or float
%% : outputs a % character

scanf(formatted string)
- similar syntax to printf()

fflush()
```

```
feof()
fclose()
```

Task 6: printf()

```
#include <stdio.h>

void main(void){
        char * emsg = "test error message";
        int lno = 27;
        printf("The sum of %d, %d, and %d is %d.\n", 65, 87, 33,
        printf("Error %s occured at line %d.\n", emsg, lno);
        printf("Hexadecimal form of %d is %x. \n", 59, 59);
        float f = 0.0314;
        printf("float f = %f, %F, %g, %G.\n", f, f, f, f);
        double d = -0.000000314359265;
        printf("double d = %e,%1.2e, %E, %g, %G. \n", d, d, d, d

}

// operation
// gcc -o task6 task6.c
// ./task6

// result
// The sum of 65, 87, and 33 is 185.
// Error test error message occured at line 27.
// Hexadecimal form of 59 is 3b.
// float f = 0.031400, 0.031400, 0.0314, 0.0314.
// double d = -3.143593e-07,-3.14e-07, -3.143593E-07, -3.14359E-
```

Task 7: scanf()

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

void main(void){
        int intValue;
        char charValue;
        char* stringValue = (char *)malloc(256 * sizeof(char));
        printf("Input intValue charValue stringValue\n");
        scanf(" %d %c %s", &intValue, &charValue, stringValue);
        printf("Got inputs intValue = %d, charValue = %c, string
}
// \"%s\" : for whitespace

// operation
// gcc -o task7 task7.c
// ./task7

// terminal
// print: Input intValue charValue stringValue
// input:10 x helloworld
// print: Got inputs intValue = 10, charValue = x, string = hell
```

Task 8: I/O re

direction use dup()

```
int dup(int oldfd)

#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```
char* cmd[] = {"ls", "-l", 0};
int main(int argc, char* argv[]){
        int fd = open(argv[1], O_WRONLY | O_CREAT, 0600);
        // fd will be 3; a file will be open in write mode
        int fd2 = dup(fd); // duplicate fd-th pointer to entry 4
        close(STDOUT_FILENO);
        dup(fd);
         // duplicate the fd-th pointer to entry 1 of the file d
        execvp(cmd[0], cmd);
        close(fd); // actually will not be executed if execvp su
        return -1;
}


// gcc -o task8 task8.c
// ./task8 tmp.txt

// result: change content of tmp.txt
// total 168
// -rwx------ 1 21094655d cstudent 8464 Nov  2 11:31 task2
// -rwx------ 1 21094655d cstudent  209 Nov  2 11:31 task2.c
// ...
```

Task 9: pipe function cal

```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>

int main(){
        int fd[2];
        int status;
```

```
        pipe(fd); // fd[0] is for reading, and fd[1] for writing
        pid_t childpid = fork();
        if (childpid == 0){
                close(fd[1]);
                char data[256];
                read(fd[0], data, sizeof(data));
                printf("%s\n", data);
                close(fd[0]);
                return 0;
        } else if (childpid > 0){
                close(fd[0]);
                char* data = "helloworld\n";
                write(fd[1], data, strlen(data)+1);
                close(fd[1]);
                wait(&status);
                return 0;
        } else { // in the parent process and error is happening
                perror("Parent: something is wrong with the chil
                return -1;
        }
}

// operation
// gcc -o task9 task9.c
// ./task9

// output:
// helloworld
```