CS 2506
review

2018/2019

Code example

- This example displays on the screen the content of a file whose name was entered by the user.
- *fopen* function is used to open a file.
- if the file is successfully opened, then fopen returns a pointer to file; if it is unable to open a file, then it returns NULL.
- *fgetc* function returns a character which is read from the file and fclose function closes the file.

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

int main()
{
        char ch, file_name[25];
        FILE *fp;
        printf("Enter the name of file whose content you wish to see\n");
        gets(file_name);
        fp = fopen(file_name,"r"); // read mode
        if( fp == NULL )
        {
                perror("Error while opening the file.\n");
                exit(EXIT_FAILURE);
        }
        printf("The contents of %s file are :\n", file_name);
        while( ( ch = fgetc(fp) ) != EOF )
                printf("%c",ch);
        fclose(fp);
        return 0;
}
```

Process lifecycle

1. Create the executable.

2. Command the execution of the program (executable file).
   – what sequence of operations is executed by the kernel?

3. Process running.
   – detail the execution of bold instructions.

4. Process terminated.
   – what does the kernel?

Step 1

- New process ➔ create its *identity* and *admin structures*.
- Allocate resources: main memory page(s).
- Insert the process in the *ready-to-execute queue*.
- The process is *scheduled* for execution.

Step 2

- Library calls ➔ system calls

- *fopen()* calls *open()*
- int open(char *path, int flags [, int mode]) makes a request to the operating system to use a file.
- The 'path' argument specifies the file to be used. The 'flags' and 'mode' arguments specify how to use it.
- If the operating system approves, it will return a file descriptor. This is a non-negative integer. Any future accesses to this file needs to provide this file descriptor.
- If it returns -1, then the access ha been denied; check the value of global variable "errno”.

Step 3

- Process execution is completed.
- What actions related to this process are taken by the kernel ?

1. OS architecture, models, concepts

- Management of computing resources, multiplication, virtualization
- Layer architecture, kernel services
- Models: hierarchical, service-based, component-based

- Process, thread, task, activities: lifecycle, events triggering state switch.

2. Process management

- Process definition, structure, lifecycle, states.
- Creating child processes
- Concurrency control
- Threads
- Process management in Linux
- Process creation in Linux

3. Scheduling

- Purpose of scheduling
- FIFO, Shortest time first
- Priority-based: multilevel feedback queues (parameters adjusting at run-time)
- Two-level scheduling
- Real-time scheduling
- Group and domain scheduling, policies
- Load balancing
- Linux scheduling

4. Memory management

- Address translation
- Virtual memory (pages, page table)
- Free space, fragmentation
- Memory allocation algorithms, over allocation/swapping
- Replacement algorithms
- Win NT page working set
- Linux memory management

5. I/O management

- I/O subsystem
- Driver's interaction and families
- Device driver structure
- Representation of devices in Unix
- I/O devices in Linux
- I/O schedulers
- Sensors

6. The file system

- File concept, structure, operations
- File system services, metadata, management
- directories
- Storage management
- Linux virtual file system: structure, main components, superblock, inode
- EXT3, disk scheduling
- Block, character devices

- What should we consider the core knowledge of this course?
  - Concepts;
  - Services;
  - Algorithms;
  - Functional elements;
  - Non-functional elements;
  - Lab skills on basic kernel functions.