National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 330: Operating Systems

BSCS:6AB

Lab 4: Memory Management

CLO3(Design & implement various pieces of OS software)

Date: 2-10-2018

Time: 10.00 AM – 1:00 PM and 02:00 PM – 05:00 PM

Instructor: Dr. Fahad Javed

Lab 4: File Systems

Introduction

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

Objectives

Understand the file structures.

Tools/Software Requirement

- gcc

Description

File structure is a structure, which is according to a required format that operating system can understand. A file has a certain defined structure according to its type. A text file is a sequence of characters organized into lines. A source file is a sequence of procedures and functions. An object file is a sequence of bytes organized into blocks that are understandable by the machine. When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

Before proceeding to task you are supposed to go through the following system calls: -

i) open
ii) read
iii) write
iv) close

Let's continue with our first system call open() whose purpose is to open file for reading or writing or to create new file. In order to demonstrate other system calls here's the example code for program that copies input file passed as first argument into output file passed as second argument:

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

#define BUF_SIZE 8192

int main(int argc, char* argv[]) {

    int input_fd, output_fd;    /* Input and output file descriptors */
    ssize_t ret_in, ret_out;    /* Number of bytes returned by read() and write() */
    char buffer[BUF_SIZE];      /* Character buffer */

    /* Are src and dest file name arguments missing */
    if(argc != 3){
        printf ("Usage: cp file1 file2");
        return 1;
    }

    /* Create input file descriptor */
    input_fd = open (argv [1], O_RDONLY);
    if (input_fd == -1) {
            perror ("open");
            return 2;
    }

    /* Create output file descriptor */
    output_fd = open(argv[2], O_WRONLY | O_CREAT, 0644);
    if(output_fd == -1){
        perror("open");
        return 3;
    }

    /* Copy process */
    while((ret_in = read (input_fd, &buffer, BUF_SIZE)) > 0){
            ret_out = write (output_fd, &buffer, (ssize_t) ret_in);
            if(ret_out != ret_in){
                /* Write error */
                perror("write");
                return 4;
            }
    }

    /* Close file descriptors */
    close (input_fd);
    close (output_fd);

    return (EXIT_SUCCESS);

}
```

Tasks
Task 1:

Write a simple grep like simple utility in the C programming language. Approach the problem by following these steps:

a) Your task is to write a simple grep like utility in the C programming language. You can name it “mygrep”. grep is basically used for searching. For example, “mygrep foo myfile” command returns all the lines that contain a string matching the expression "foo" in the file. Your code will be checked by running such a simple command.

   It is easy to get overwhelmed with the complexity of grep. Note that we are looking for a simple version that can search for a specific string in a single file. You may consult grep man page if you are looking to get more info about it

Deliverables:

Submit your code as well as README file. In your README, briefly describe how your algorithm works.

Submission Guideline:
Submit your source code in a document and instead of submitting separate README file write down the required explanation in your document with the heading README.