CPE 460 Laboratory 7: File I/O

Department of Computer Engineering
Yarmouk University

Spring 2017

1 Purpose

Understand the basics of File I/O System calls of UNIX

```
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
int creat(const char *pathname, mode_t mode);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
int close(int fd);
```

2 Introduction

What happens when a process opens a file? A process may have several open files which it may be reading from and writing to. It also has a current position within the file, which is the next byte to be read or written. Each process has its own array to keep track of:

1. The file opened.
2. The files status (i.e, whether open for reading or writing)
3. The current offset within the file

When a file is opened or created by a process the kernel assigns a position in the array called the file descriptor. Each entry of this array actually contains a pointer to a file table which stores each of the three pieces of information: file, file status flags, and offset. The file table does not itself contain the file, but instead has a pointer to another table (called the `vnode` table), which has vital information about the file, including its location in memory.

3 Exercise

1. Given a file that consists of one line containing a single word. Your task is to write a program that reads that word from a file.
2. Write a program to read a set of shell commands written into a file named cmd.sh, where each command is written in a separate line. The program reads the file line by line, forks a child for each command (line) it reads.
   Note: The parent process needs to wait until the child executes the command using `execlp` and `wait` system calls.