Operating Systems

CMPT 424 • Fall 2020

*i*Project Four (final) - 200 points

Goals

To improve on the functionality from *i*Project Three (all of which is required) by adding a local file system and swapped virtual memory so you can execute more processes than you have partitions for in memory. Also, to make something of which you are proud, that you can show people, and brag about, and talk about in job interviews for years to come.

Functional Requirements

Add shell commands for the following disk operations:

- ☐ `create <filename>` — Create the file *filename* and display a message denoting success or failure. [10 points]
- ☐ `read <filename>` — Read and display the contents of *filename* or display an error if something went wrong. [10 points]
- ☐ `write <filename> "data"` — Write the data inside the quotes to *filename* and display a message denoting success or failure. [10 points]
- ☐ `delete <filename>` — Remove *filename* from storage and display a message denoting success or failure. [10 points]
- ☐ `format` — Initialize all blocks in all sectors in all tracks and display a message denoting success or failure. [10 points]

- ☐ Add a shell command, `ls`, to list the files currently stored on the disk. [5 points]
- ☐ Add a shell command to allow the user to select a CPU scheduling algorithm — `setschedule [rr, fcfs, priority]` [3 points]
- ☐ Add a shell command, `getschedule`, to return the currently selected cpu scheduling algorithm. [2 points]
- ☐ [challenge] See challenges on next page [+ points]

Implementation Requirements

- ☐ Implement a file system in HTML5 session storage as discussed in class. [20 points]
- ☐ Include a file system viewer in your OS interface. [5 points]

Develop a File System Device Driver (fsDD) for all of the functional requirements noted above.

- ☐ Load the fsDD in a similar manner as the keyboard device driver. [5 points]
- ☐ Develop your fsDD to insulate and encapsulate the implementation of the kernel-level I/O operations (noted above) from the byte-level details of your individual blocks on the local storage. [20 points]

Add new scheduling algorithms to your CPU scheduler. Default to RR.

- ☐ first-come, first-served (FCFS) [10 points]
- ☐ non-preemptive priority (You will need an optional load parameter here.) [10 points]

Implement swapped virtual memory with enough physical memory for three concurrent user processes.

- ☐ Allow the OS to execute four or more concurrent user process by writing roll-out and roll-in routines to . . . [70 points]
  - Take a ready process and store it to the disk via your fsDD.
  - Load a swapped-out process from disk and put it in the ready queue.
  - Your ready queue should denote which processes are where.
- ☐ Your code must separate structure from presentation, be professionally formatted, use and demonstrate best practices, and make me proud to be your teacher. [−∞ if not]
- ☐ You must commit to Git early and often. I am not kidding. [−∞ if not]

Submitting

Update GitHub with your current code. Tell me what branch to grade.

Operating Systems

CMPT 424 • Fall 2020

Challenges: [points vary with impressiveness]

- ☐ Implement command line options for format:
  - `format -quick` : initialize the first four bytes of every directory and data block
  - `format -full` : same as quick and also initializes bytes 4-63 in directory and data blocks too.
- ☐ Support hidden files (that do not show up in `ls` output) with filenames that begin with a period.
- ☐ Implement a command line option for ls:
  `ls -l` : lists all file names (even hidden ones) as well as their size and create date.
- ☐ Implement chkdsk and fsck-like utilities:
  - recover deleted files
  - reclaim data blocks that are in use but not indexed in the directory blocks
  - defragment the data blocks
- ☐ Implement file `copy` and `rename` commands.