Operating Systems

CMPT 424 • Fall 2024

– *i*Project Four (final) - 150 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 and implement functionality for the following disk operations, displaying a message of success or failure for each one:

- ☐ `format` — Initialize all blocks in all sectors in all tracks [7 points]
- ☐ `create <filename>` — Create the file *filename* [7 points]
- ☐ `read <filename>` — Read and display the contents of *filename* [7 points]
- ☐ `write <filename> "data"` — Write the data inside the quotes (but not the quotes themselves) to *filename* [7 points]
- ☐ `delete <filename>` — Remove *filename* from storage [7 points]
- ☐ `copy <existing filename> <new filename>` — copy [7 points]
- ☐ `rename <current filename> <new filename>` — rename [7 points]
- ☐ `ls`— list the files currently stored on the disk. [7 points]

Challenges: See many challenges on next page. [+ points]

Implementation Requirements

Implement your file system in HTML5 session storage as discussed in class, including

- ☐ a disk system viewer in your OS interface [7 points]
- ☐ a Disk System Device Driver (dsDD) for all of the functional requirements noted above. [7 points]
  - Load the dsDD in a similar manner as the keyboard device driver.
  - Develop your dsDD 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.

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 . . . [80 points]
  - Take a ready process and store it to the disk via your dsDD.
  - 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. By the end of this semester you should have over 100 commits.

Submitting

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

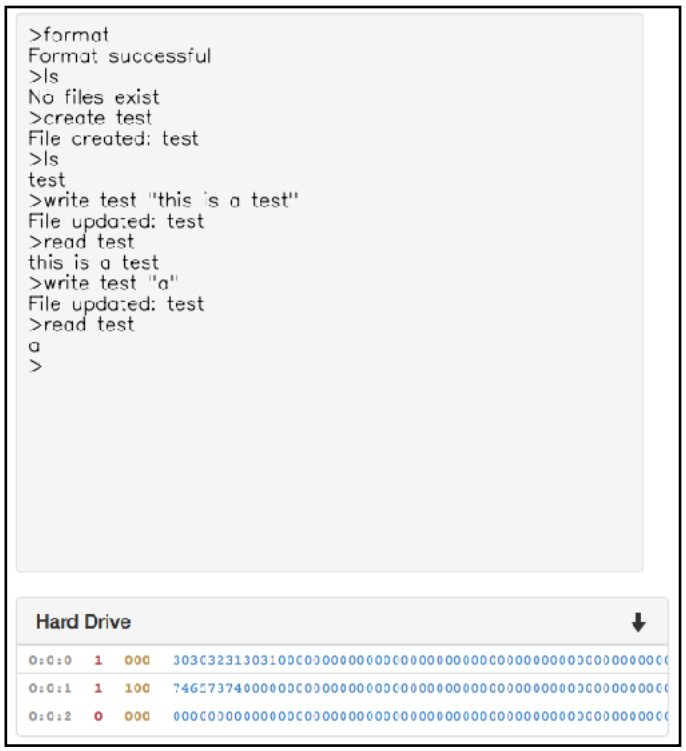
© 2008-2112 Alan G. Labouseur, All Rights Reserved

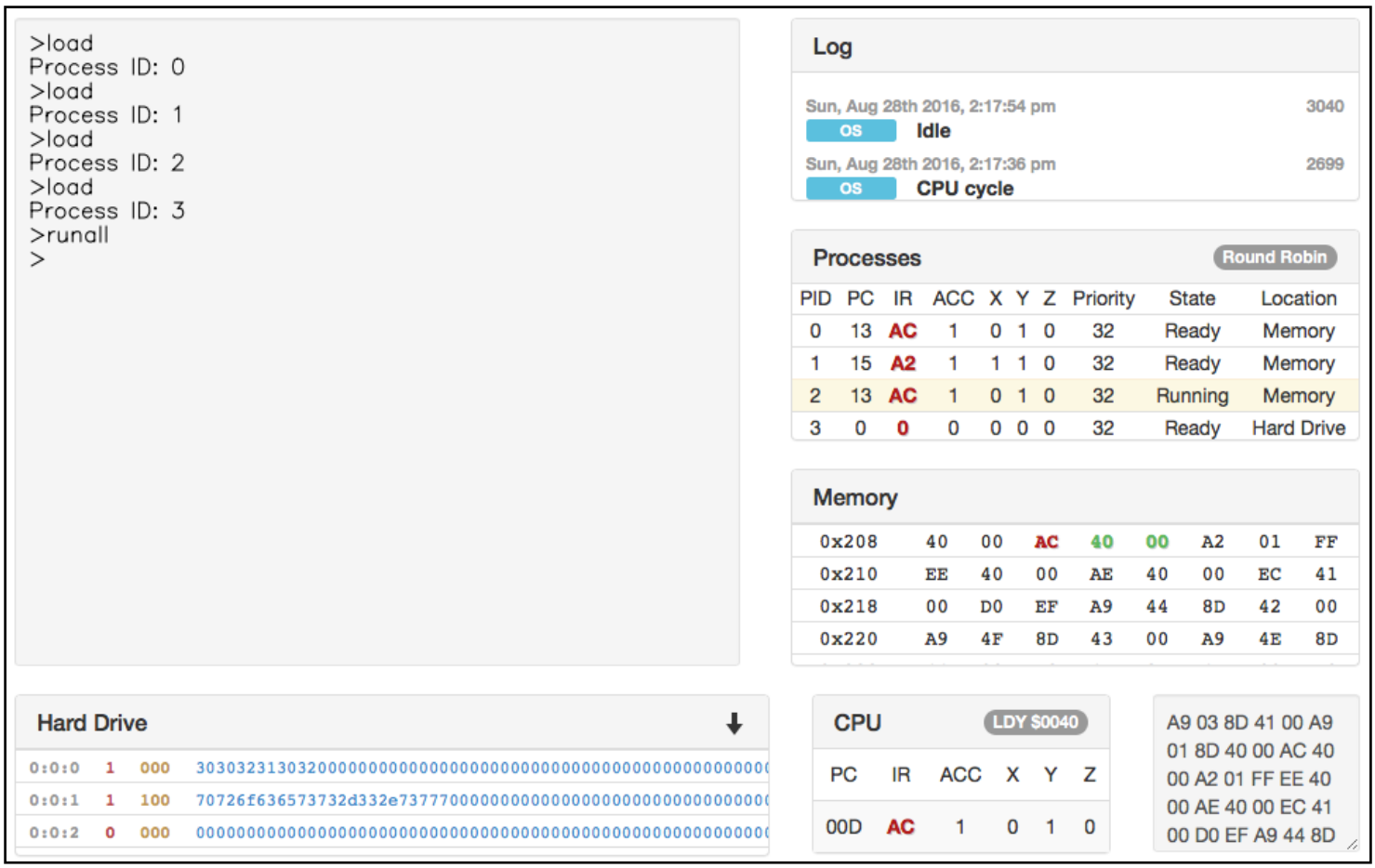
Operating Systems

CMPT 424 • Fall 2024

Challenges: [+points vary with impressiveness]

- ☐ `format -quick` : Initialize just the first four bytes of every directory and data block
- ☐ 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 -a` that lists all file names (even hidden ones) as well as their size and create date.
- ☐ `link file1 file2` : make both file names point to the same file data. (What happens on delete?)
- ☐ 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
- ☐ `alias command1 command2` : Alias shell commands so you can invoke with either one.
- ☐ Add FCFS and non-preemptive priority scheduling algorithms to your CPU scheduler. (Keep RR as the default.) Include `getSchedule` and





© 2008-2112 Alan G. Labouseur, All Rights Reserved