Operating Systems
CMPT 424 • Fall 2021

Project One - 100 points

Goals

To review and come to intimately understand our initial project as well as the awesomeness that is TypeScript. Then to add all the functionality specified below.

☐ Alter the ver command to display your own data. [5 points]
☐ Add some new shell commands:
  • date - displays the current date and time
  • whereami - displays the users current location (use your imagination)
  • something else interesting and creative; surprise me [15 points]
☐ Enhance the host display with a graphic task bar that displays . . .
  • the current date and time
  • status messages as specified by the user with a new shell command:
    status <string>
    example: status I love Operating Systems
[10 points]
☐ Implement scrolling in the client OS console/CLI. [30 points]
☐ Other console/CLI enhancements:
  • Accept and display punctuation characters and symbols.
  • Handle backspace appropriately.
  • Implement command completion with the tab key.
  • Provide command history recall via the up and down arrow keys. [30 points]
☐ Display a BSOD message (on the CLI) when the kernel traps an OS error.
  • Add a shell command to test this. Remember to include it in the help. [5 points]
☐ Add a shell command called load to validate the user code in the HTML5 text area (id= “taProgramInput”). Only hex digits and spaces are valid. [5 points]
☐ [challenge] Implement line-wrap in the CLI. [+10 points]

☐ Your code must ...
  • separate structure from presentation.
  • be professionally formatted.
  • use and demonstrate best practices.
  • make me proud to be your teacher. [−∞ if not]

☐ Do not break GLaDOS. Don’t make me flood the Enrichment Center with deadly neurotoxin. Again. (For science.)

Implementation Requirements

Functional Requirements

General Hints

Read up on the Canvas before you mess with the console/CLI. There are some helpful links on our class web site about the HTML5 canvas. Do some Canvas experiments on your own. It’s really quite amazing what you can do with it.

Remember the utility of comments and how much their presence and quality effect my opinion of your work. Also, write code that is uniquely you.

Make many commits to Git. I do not want to see one massive “everything” commit when I review your code. (It’s −∞ if you do that.) Commit early and often. And make sure your commit messages are descriptive, informative, and — if possible — entertaining.
Specific Hints

• Keep .js and .ts files in separate directories.
• An empty text area means the source code is not valid.
• Scrolling is difficult. Think carefully. And do not scroll the entire canvas. You must scroll the text within the canvas, but it only has to scroll forward.
• Regarding punctuation characters: & is not the same as ↑.
• Command completion with the tab key can be tricky if there is more than a single match for the letters typed before pressing tab. Think carefully about how you handle that. I want to see elegant solutions.
• Be sure to add a .gitignore file so your your IDE configuration files and other messy stuff are excluded from your Git repository.

Submitting Your Work

Add me (username Labouseur) as a collaborator to your private GitHub repository. Then e-mail me the URL. Send this to me before the beginning of the class in which this is due.

Note: Your project will not be accepted for grading unless and until your repo is private and you have added me as a collaborator.

>help
Commands:
   ver – Displays the current version data
   help – Lists all available commands
   shutdown – Shuts down SvegOS
   cls – Clears the screen
   man <topic> – Displays the manual page for <topic>
   trace <on | off> – Enables/disables the OS trace
   rot13 <string> – Does rot13 encryption on <string>
   quantum <integer> – Changes the CPU quantum
   prompt <string> – Sets the prompt
   date – Displays the current date and time
   whereami – Displays the current location of the user
   status <string> – Sets a status message
   ps – Shows all active processes
   kill <integer> – Terminates the specified process
   load [ [<priority>] ] – Loads the specified user program
   \ <regex> <function> – Filters function output
   bsd – Enables the blue screen of death
   run <processid> – Executes a program in memory
   runall – Executes all programs
   create <filename> – Creates the specified file
   read <filename> – Reads the specified file
   write <filename> "data" – Writes the specified file
   delete <filename> – Deletes the specified file
   format – Initializes disk
   ls – Lists all files on disk