Assignment 2

We'll be doing a fair bit of work on the command line this trimester, so this assignment is directed toward learning about the UNIX/Linux shell. There's a very good treatment of the shell in MIT's *The Missing Semester of Your CS Education* lecture series. If you're a complete shell novice, you may find the Software Carpentry course *The UNIX Shell* https://swcarpentry.github.io/shell-novice/ more accessible. Both of these resources can be used as references for this assignment. You may also use AI tools for research, but beware that they won't always give answers meeting the requirements of the question!

Submit your answers in PDF to the ECS submission system https://apps.ecs.vuw.ac.nz/submit/ENGR301. You are welcome to use the Markdown template for Assignment 2 in writing your answers.

Core Questions

Complete the following simple questions, assuming a context of zsh on the ECS workstations. These questions require only a short sentence or two to answer.

1. [5%] Most shells keep a list of previously executed commands, known as the history. What is the zsh command to display the history on the screen?
2. [5%] The history and other output can often be larger than the terminal display lines. In one sentence explain what a "pager" is in zsh and name the most commonly used pager.
3. [10%] what is the command using a "pipe" to use a pager with the output of the command displaying the history?
4. [10%] Environment variables allow configuration of the shell behaviour. What are the environment variables to set the default editor in zsh?
5. [10%] To make an environment variable setting persist between sessions, where should you place the setting in zsh?

Completion Questions

Complete the following simple questions, assuming a context of zsh on the ECS workstations. These questions require only a short sentence or two to answer.

1. [10%] The Emacs keybindings (keyboard shortcuts) in bash and zsh allow quick navigation when constructing commands. What are the keyboard shortcuts for: previous-command, beginning-of-line, end-of-line, forward-word, backward-word, delete-character (forward), kill-word, kill-line, yank (paste)

and rotate through the yank list.

2. [10%] What is the Emacs keybinding to reverse interactive search the history from the command line *and* what is a keybinding (keystroke) which will exit from interactive search mode?

3. [10%] Everyone should have set up SSH keys and cloned the data-recorder repository, but if you set a passphrase on your SSH key (you should have!) then git will still ask for the passphrase every time you interact with the repository. In one sentence each, what is `ssh-agent` and what does `ssh-add` do?

Challenge Questions

These questions are a little more challenging and will likely require some experimentation in the shell. In each case give the simplest possible command (i.e. using the fewest flags) which meets the task. It's preferable to use the long forms of flags, if available, in your answers.

1. [10%] Construct a single `find` command which finds all Python (`*.py`) files in the `main` branch of your data-recorder repository and then executes a `grep` command on the found files which prints the shebang (`#!`) line (only where present). The output should look like:

```
./path/to/file.py:#!/usr/bin/env python
...
```

2. [10%] The data recorder output log looks like the example appended. To analyse the occurrence of the different wake reasons, what is the `grep` command to show just the text `machine.wake_reason(): ...` to the end of the line, i.e. including the digit?

   Hint: read the manual page (`man grep`) to find the flag to only show the matching text and use the short regular expression `.*` in your `grep` match string to match any string of characters. You might find it helpful to save the sample log above to a file to test.

3. [10%] Log analysis with `grep` can generate lots of lines, potentially 10's of thousands. The commands `sort` and `uniq --count` (see `man uniq` for details of this command and its flag) can help with the analysis. For example, we might be interested in the total count of `wake_reason` s in the log to troubleshoot a problem. For the sample log below, the output should look like:

```
<count> machine.wake_reason(): <code>
<count> machine.wake_reason(): <code>
```

   What is the simplest one-line expression which connects the output of `grep` command from the question above to `sort` and `uniq --count` to produce output in the above format *and* sort the output in descending order by count?

Sample Log Output

```
2024-03-08 16:00:08  DEBUG  main: machine.wake_reason(): 0
2024-03-08 16:00:08  DEBUG  main: machine.reset_cause(): 3
2024-03-08 16:00:08  INFO   main: Entering Regular Mode
...
2024-03-08 16:00:22  INFO   main: Entering deep sleep for 578 s (9.6 minutes)
2024-03-08 16:10:08  DEBUG  main: machine.wake_reason(): 4
2024-03-08 16:10:08  DEBUG  main: machine.reset_cause(): 4
2024-03-08 16:10:08  INFO   main: Entering Regular Mode
...
2024-03-08 16:10:22  INFO   main: Entering deep sleep for 578 s (9.6 minutes)
...
2024-03-08 16:10:08  DEBUG  main: machine.wake_reason(): 4
2024-03-08 16:10:08  DEBUG  main: machine.reset_cause(): 4
2024-03-08 16:10:08  INFO   main: Entering Regular Mode
```