

MiniShell

Deadline: October 31st, 2014. **For this assignment you may work in pairs.**

1 Instructions

You must implement a program `minish` which can be used as a Unix shell.

`minish` must support at least simple commands with zero or more arguments, and commands separated by semicolons. Extra whitespace must be condensed. For example `echo hello ; echo world` must behave the same as `echo hello;echo world`. The shell must also support the built-in commands `exit` and `cd`; both with either zero or one argument.

You must then choose a selection of features in the following set to achieve a passing grade and beyond, see [Grading](#) below:

- Command pipelines, eg. `ls | grep foo`
- Simple redirections at the end of simple commands, eg. `tr a-z A-Z <fin >fout`
- File globbing: support for expanding `*`, `?` etc. in a command line into a list of files/directories using `glob(3)`.
- Setting or unsetting environment variables with `setenv` and `unsetenv`. (Note: you need not implement variable expansion. If you do, it must be unsurprising.)
- Simple quoting: `"foo bar"` is passed as a single argument. Must support simple `\` escapes: `"foo\"bar"` counts as one argument with 7 characters (`foo"bar`). If globbing is supported, quoting must disable globbing, ie. `"*` is not expanded and passed as a single character.
- Command groups enclosed in `{...}` (same shell) or `(...)` (sub-shell).
- Job control: `command &` detaches; `Ctrl+Z` suspends, `jobs` lists the currently managed jobs; `fg/bg` resumes a job in the foreground or background.
- Any extra feature you deem particularly useful.

Constraints:

- You must only depend on standard C functions (either from ISO C 1999/2011 or POSIX).
- You must not use `system` or any other mechanism that invokes an existing shell to execute the command lines.

2 Example commands

```
$ echo hello; touch /tmp/hello.txt
hello
$ ls /tmp | grep .txt
hello.txt
$ ls /dev >/tmp/hello.txt
$ tr d @ </tmp/hello.txt | grep st@
st@err
st@in
st@out

# Example globbing:
$ ls /dev/std* /tmp/*.c

# Example quoting:
$ echo "hello;\n world"
hello;n world

# Example environment variables:
$ setenv MY hello
$ bash -c "echo $MY"
hello

# Example command groups:
$ ( echo hello; echo world ) >hello.txt
$ tr \n . <hello.txt
hello.world

$ setenv I hello; { setenv I world }; bash -c "echo $I"
world
$ setenv I hello; ( setenv I world ); bash -c "echo $I"
hello
```

3 Grading

- 3 points if your shell properly handles simple commands separated by semicolons, and if the built-ins `exit` and `cd` are properly implemented.
- +2 points if command pipelines are properly supported.
- +1 point if simple redirections are properly supported.
- +1 point if `setenv/unsetenv` are properly supported.
- +1 point if simple quoting is properly supported.
- +2 points if file globbing is properly supported.
- +2 points if command groups are properly supported.
- +4 points if job control is properly implemented.
- +2 point if any additional feature is implemented and duly documented in an enclosed `README` file, and your examiner agrees it is indeed useful.

Tips: the maximum grade is 10; there are multiple ways to get there. As usual, go for correctness before completeness. Test a lot. Use version control. Consider pair programming.

4 Copyright and licensing

Copyright © 2014, Raphael 'kena' Poss. Permission is granted to distribute, reuse and modify this document and other documents for the Systems Programming course by the same author according to the terms of the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>.