POSIX Command Line Utilities

Just A Glimpse

Nov 22nd, 2015

Siyuan Liu

leasunhy@gmail.com

Microsoft Student Technology Club

Sun Yat-sen University

Agenda

- **POSIX**: **P**ortable **O**perating **S**ystem **I**nterface

- **POSIX**: **P**ortable **O**perating **S**ystem **I**nterface
- **CLI**: **C**ommand **L**ine **I**nterface

- **POSIX**: **P**ortable **O**perating **S**ystem **I**nterface
- **CLI**: **C**ommand **L**ine **I**nterface
- Part of **POSIX** are CLI utilities, the subject today.

- **POSIX**: **P**ortable **O**perating **S**ystem **I**nterface
- **CLI**: **C**ommand **L**ine **I**nterface
- Part of **POSIX** are CLI utilities, the subject today.
- **kernel**: The kernel of OS is a program that manages resources and implement IPC.

- **POSIX**: **P**ortable **O**perating **S**ystem **I**nterface
- **CLI**: **C**ommand **L**ine **I**nterface
- Part of **POSIX** are CLI utilities, the subject today.
- **kernel**: The kernel of OS is a program that manages resources and implement IPC.
- **shell**: Opposite of kernel. The machine/human interface.

- Solve various tasks efficiently.

- Solve various tasks efficiently.
- Solve programming-needed tasks, without programming.

- Solve various tasks efficiently.
- Solve programming-needed tasks, without programming.
- Functional programming like experience: simple utilities pipelined to solve complex problems.

- Solve various tasks efficiently.
- Solve programming-needed tasks, without programming.
- Functional programming like experience: simple utilities pipelined to solve complex problems.
- Most servers don't have a GUI, so CLI may be your only choice.

- Solve various tasks efficiently.
- Solve programming-needed tasks, without programming.
- Functional programming like experience: simple utilities pipelined to solve complex problems.
- Most servers don't have a GUI, so CLI may be your only choice.
- **Bigger** than **bigger**.

- Solve various tasks efficiently.
- Solve programming-needed tasks, without programming.
- Functional programming like experience: simple utilities pipelined to solve complex problems.
- Most servers don't have a GUI, so CLI may be your only choice.
- **Bigger** than **bigger**.
- ... Just bacause it's a lot of fun :-)

- `cd`, `pwd`, `ls`

- `cd`, `pwd`, `ls`
- `cp`, `mv`, `rm`

- `cd`, `pwd`, `ls`
- `cp`, `mv`, `rm`
- `echo`

- `cd`, `pwd`, `ls`
- `cp`, `mv`, `rm`
- `echo`
- Running an executable.

- `cd`, `pwd`, `ls`
- `cp`, `mv`, `rm`
- `echo`
- Running an executable.
- Interruptting a running command: `Ctrl+C`.

- `cd`, `pwd`, `ls`
- `cp`, `mv`, `rm`
- `echo`
- Running an executable.
- Interruptting a running command: `Ctrl+C`.
- Input an `End Of File`: `Ctrl+D`.

- `cd`, `pwd`, `ls`
- `cp`, `mv`, `rm`
- `echo`
- Running an executable.
- Interruptting a running command: `Ctrl+C`.
- Input an `End Of File`: `Ctrl+D`.
- Suspending a running command: `Ctrl+Z`.

- `cd`, `pwd`, `ls`
- `cp`, `mv`, `rm`
- `echo`
- Running an executable.
- Interruptting a running command: `Ctrl+C`.
- Input an `End Of File`: `Ctrl+D`.
- Suspending a running command: `Ctrl+Z`.
- ... and resuming: `fg`.

- `ls /etc` lists the contents of `/etc`

- `ls /etc` lists the contents of `/etc`
- `...` where `ls` is the name of the command,

- `ls /etc` lists the contents of `/etc`
- `...` where `ls` is the name of the command,
- `...` and `/etc` is the **SECOND** argument (argv[1]).

- `ls /etc` lists the contents of `/etc`
- ... where `ls` is the name of the command,
- ... and `/etc` is the **SECOND** argument (argv[1]).
- Where has the **FIRST** argument gone?

- `ls /etc` lists the contents of `/etc`
- ... where `ls` is the name of the command,
- ... and `/etc` is the **SECOND** argument (argv[1]).
- Where has the **FIRST** argument gone?
- It's the name of the executable:
  here, the first argument is `ls`.

- Completion: `Tab`

- Completion: `Tab`
- Fast cursor movement:
  `Ctrl/Alt+F`, `Ctrl/Alt+B`, `Ctrl+A/E`

- Completion: `Tab`
- Fast cursor movement: `Ctrl/Alt+F`, `Ctrl/Alt+B`, `Ctrl+A/E`
- Repeat: `Alt+A`.

- Completion: `Tab`
- Fast cursor movement: `Ctrl/Alt+F`, `Ctrl/Alt+B`, `Ctrl+A/E`
- Repeat: `Alt+A`.
- Clear: `Ctrl+L`

Shell Basics
Key Bindings (Emacs-like)

- Completion: `Tab`
- Fast cursor movement:
  `Ctrl/Alt+F`, `Ctrl/Alt+B`, `Ctrl+A/E`
- Repeat: `Alt+A`.
- Clear: `Ctrl+L`
- Fast line editing:
  `Alt+D / Ctrl+W`, `Alt+W / Ctrl+K`, `Ctrl+H/D`

- Completion: `Tab`
- Fast cursor movement:
  `Ctrl/Alt+F`, `Ctrl/Alt+B`, `Ctrl+A/E`
- Repeat: `Alt+A`.
- Clear: `Ctrl+L`
- Fast line editing:
  `Alt+D / Ctrl+W`, `Alt+W / Ctrl+K`, `Ctrl+H/D`
- Reverse search: `Ctrl+R`
  – it is awesome!

- $*$: matches anything, greedily.

- `*`: matches anything, greedily.
- `{}`:
  `abc{1, 2}` expands into `abc1 abc2`
  `abc{1..10}` expands into `abc1 abc2 abc3 abc4 ... abc10`

- Example: `echo 1 2 | ./a.out`
- Example: `./a.out` and then input `1 2`

- >, >>
- <
- Special: <<.

- `/dev/stdin`: stdin. Any explanation needed?

- `/dev/stdin`: stdin. Any explanation needed?
- `/dev/stdout`: stdout. ?

- `/dev/stdin`: stdin. Any explanation needed?
- `/dev/stdout`: stdout. ?
- `/dev/zero`, `/dev/one`: zero and one.

- `/dev/stdin`: stdin. Any explanation needed?
- `/dev/stdout`: stdout. ?
- `/dev/zero`, `/dev/one`: zero and one.
- `/proc/meminfo`, `/proc/cpuinfo`: Memory/CPU information.

- `/dev/stdin`: stdin. Any explanation needed?
- `/dev/stdout`: stdout. ?
- `/dev/zero`, `/dev/one`: zero and one.
- `/proc/meminfo`, `/proc/cpuinfo`: Memory/CPU information.
- `/proc/<id:int>`: Information of the process with pid `id`.

- `/dev/stdin`: stdin. Any explanation needed?
- `/dev/stdout`: stdout. ?
- `/dev/zero`, `/dev/one`: zero and one.
- `/proc/meminfo`, `/proc/cpuinfo`: Memory/CPU information.
- `/proc/<id:int>`: Information of the process with pid `id`.
- `/dev/null`: BLACK HOLE.

- What black hole?

- What black hole?
- Why black hole?

Shell Basics

/dev/null: Black Hole Explained

- What black hole?
- Why black hole?
- Example: `cat > /dev/null`

- What black hole?
- Why black hole?
- Example: `cat > /dev/null`
- Jokes:

- What black hole?
- Why black hole?
- Example: `cat > /dev/null`
- Jokes:
- Please send complains to /dev/null.

- What black hole?
- Why black hole?
- Example: `cat > /dev/null`
- Jokes:
- Please send complains to /dev/null.
- I've learned all my courses and remembered them using my /dev/null.

- What black hole?
- Why black hole?
- Example: `cat > /dev/null`
- Jokes:
- Please send complains to /dev/null.
- I've learned all my courses and remembered them using my /dev/null.
- I surely saved my money from 11.11. I put them in my /dev/null.

- `man`: Read manual for the command.

- `man`: Read manual for the command.
- `apropos`: Search manual page title and description.

- `man`: Read manual for the command.
- `apropos`: Search manual page title and description.
- `awesome-command --help`: Show help for `awesome-command`.

Basic Utilities

`cat` - concatenate file(s)

- Example: `cat 1.txt`

- Example: `cat 1.txt`
- Example: `cat 1.txt 2.txt`

Basic Utilities

`cat` - concatenate file(s)

- Example: `cat 1.txt`
- Example: `cat 1.txt 2.txt`
- Use case: stay tuned.

- Example: `paste 1.txt 2.txt`

Basic Utilities

`paste` - join lines of files

- Example: `paste 1.txt 2.txt`
- Use case: `paste accounts passwords`

Basic Utilites

`head` & `tail` - first/last n lines of file

- Example: `head longlongfile.log`

Basic Utilites

`head` & `tail` - first/last n lines of file

- Example: `head longlongfile.log`
- Example: `head -n 20 longlongfile.log`

Basic Utilites

`head` & `tail` - first/last n lines of file

- Example: `head longlongfile.log`
- Example: `head -n 20 longlongfile.log`
- Example: `tail -f takeslongtime.log`

- Example: `sort accounts`

Basic Utilites

`sort` - sort lines of file

- Example: `sort accounts`
- Example: `sort -n nums`

Basic Utilites

`sort` - sort lines of file

- Example: `sort accounts`
- Example: `sort -n nums`
- Practice:
  `du -d 1` can be used to find disk usage of every subdirectory in current working directory.
  So, how can I find out the most space-consuming folder?

Basic Utilites

`sort` - sort lines of file

- Example: `sort accounts`
- Example: `sort -n nums`
- Practice:
  `du -d 1` can be used to find disk usage of every subdirectory in current working directory.
  So, how can I find out the most space-consuming folder?
- Answer: `du -d 1 | sort -n | tail -n 2`

Basic Utilites

`sort` - sort lines of file

- Example: `sort accounts`
- Example: `sort -n nums`
- Practice:
  `du -d 1` can be used to find disk usage of every subdirectory in current working directory.
  So, how can I find out the most space-consuming folder?
- Answer: `du -d 1 | sort -n | tail -n 2`
- How can you achieve the same using GUI, for instance, on Windows?

Basic Utilites

`sort` - sort lines of file

- Example: `sort accounts`
- Example: `sort -n nums`
- Practice:
  `du -d 1` can be used to find disk usage of every subdirectory in current working directory.
  So, how can I find out the most space-consuming folder?
- Answer: `du -d 1 | sort -n | tail -n 2`
- How can you achieve the same using GUI, for instance, on Windows?
- Maybe **grouping**? $O(1)$ vs $O(logn)$:-)

- Example: `uniq dup.txt`

Basic Utilites

wc - count words/lines of files

- Example: `wc -l cool.cpp`

Practise Time :-)
Problem 1

- Problem Statement:
  I'm in a folder, which contains numerous C++ source files.
  I want to know how many lines of C++ code reside in this folder.

Practise Time :-)
Problem 1

- Problem Statement:
  I'm in a folder, which contains numerous C++ source files.
  I want to know how many lines of C++ code reside in this folder.
- Answer: `wc -l *.cpp`

Practise Time :-)

Problem 2

- Problem Statement:
  For some reason (explained later), I want to concatenate all the c++ files in the current folder into a single file.

Practise Time :-)

Problem 2

- Problem Statement:
  For some reason (explained later), I want to concatenate all the c++ files in the current folder into a single file.
- Answer: `cat *.cpp > total.cpp`

Intermidiate Utilities

`xargs` - build and execute commands from stdin

- Example: `echo 1 2 3 | xargs echo`

Intermidiate Utilities

`xargs` - build and execute commands from stdin

- Example: `echo 1 2 3 | xargs echo`
- Example: `echo 1 2 3 | xargs -n 1 echo`

Intermidiate Utilities

`xargs` - build and execute commands from stdin

- Example: `echo 1 2 3 | xargs echo`
- Example: `echo 1 2 3 | xargs -n 1 echo`
- Example: `echo 1 2 3 | xargs -P 2 -n 1 echo`

Intermidiate Utilities

`find` - search for files in a directory hierarchy

- Example: `find . -type f -name \*.cpp`

Intermidiate Utilities

`find` - search for files in a directory hierarchy

- Example: `find . -type f -name \*.cpp`
- Example: `find . -type d -iname \*a\* -exec ls {}`

Intermidiate Utilities

`find` - search for files in a directory hierarchy

- Example: `find . -type f -name \*.cpp`
- Example: `find . -type d -iname \*a\* -exec ls {}`
- Practise:
  I want to know how many lines of python code I wrote for a project.
  Now I'm in `/`, and the project is in `~/Documents/project/SYSU-Software-2015/`.
  What should I do?

Intermidiate Utilities

`find` - search for files in a directory hierarchy

- Example: `find . -type f -name \*.cpp`
- Example: `find . -type d -iname \*a\* -exec ls {}`
- Practise:
  I want to know how many lines of python code I wrote for a project.
  Now I'm in `/`, and the project is in `~/Documents/project/SYSU-Software-2015/`.
  What should I do?
- Answer: `find ~/Documents/project/SYSU-Software-2015/ -type f -name \*.py | wc -l`

Intermidiate Utilities

grep - find matching lines in files

- Example: `grep iostream a.cpp`

Intermidiate Utilities

grep - find matching lines in files

- Example: `grep iostream a.cpp`
- Example: `grep -n vector a.cpp`

Intermidiate Utilities

grep - find matching lines in files

- Example: `grep iostream a.cpp`
- Example: `grep -n vector a.cpp`
- Example: `find . -type f -name \*.py | xargs grep -ni taskhall`

Intermidiate Utilities

grep - find matching lines in files

- Example: `grep iostream a.cpp`
- Example: `grep -n vector a.cpp`
- Example: `find . -type f -name \*.py | xargs grep -ni taskhall`
- Example: `grep --include=*.py -rni taskhall`

Intermidiate Utilities

grep - find matching lines in files

- Example: `grep iostream a.cpp`
- Example: `grep -n vector a.cpp`
- Example: `find . -type f -name \*.py | xargs grep -ni taskhall`
- Example: `grep --include=*.py -rni taskhall`
- Example: `grep -rne '\d+'`

- Example: `sed -i -- 's/127.0.0.1/10.1.10.1/g' *.conf`

Intermidiate Utilities

`sed` - line based text processing

- Example: `sed -i -- 's/127.0.0.1/10.1.10.1/g' *.conf`
- Example: `sed -- '/^##/a\%666' posix_cli_tools.md`

- Example: `sed -i -- 's/127.0.0.1/10.1.10.1/g' *.conf`
- Example: `sed -- '/^##/a\%666' posix_cli_tools.md`
- Example: `sed -- '/^##/i\%%%%%' posix_cli_tools.md`

- Example: `diff a.txt b.txt`
- Example: `diff 1.dir 2.dir`

- Example: `patch orig patch`

- The tools I am going to introduce to you are not part of `POSIX`.
- These great tools are not `*nix`-limited. `Windows` have them too!
- They all require some time to learn. Fear not, the time you spent on them will pay.

- Among numerous choices of editors, vim and emacs have distinguished themselves from the others.

- Among numerous choices of editors, vim and emacs have distinguished themselves from the others.
- vim: The God of Editor.

- Among numerous choices of editors, `vim` and `emacs` have distinguished themselves from the others.
- `vim`: The God of Editor.
- `emacs`: The Editor of God.

- Among numerous choices of editors, `vim` and `emacs` have distinguished themselves from the others.
- `vim`: The God of Editor.
- `emacs`: The Editor of God.
- Demostration.

- TeX is the most widespread, most advanced and most professional typesetting system.
- LaTeX handles content side of document, while TeX handles the layout side.
  Together, marvelous documents are made.

- Ever fancied conversion between `docx` and `html`, `markdown` and `latex`, even `markdown` and `h5 slides`?

- Ever fancied conversion between `docx` and `html`, `markdown` and `latex`, even `markdown` and `h5 slides`?
- `pandoc` comes to rescue!

- Ever fancied conversion between `docx` and `html`, `markdown` and `latex`, even `markdown` and `h5 slides`?
- `pandoc` comes to rescue!
- **These** slides are converted from `markdown` :-)

- Use git, use git, use git!

- Use `git`, use `git`, use `git`!
- Demonstration.

Conclusion

- To learn them, use them to solve real problems (and for fun).

Conclusion

- To learn them, use them to solve real problems (and for fun).
- Never feel ashamed for forgetting the command line options of a command. Just use `-help` or `man` when you need reference.

Conclusion

- To learn them, use them to solve real problems (and for fun).
- Never feel ashamed for forgetting the command line options of a command. Just use `-help` or `man` when you need reference.
- Have fun with CLI!

Thank you!

Brought to you by *Siyuan Liu*, MSTC, SYSU.

Nov 22nd, 2015