(Don't Fear) The Command Line

Outline for presentation to Metro Detroit Linux User Group

I. The command line interface- What?
- **A.** Where does it come from?
  - **1.** Grub boot environment, strictly speaking, is a CLI
  - **2.** So is login console
    - **a.** Login prompt
    - **b.** Password prompt
  - **3.** Bash, or other command shell started by lower level program.
    - **a.** mingetty or other getty program, usually at system start
    - **b.** xterm, konsole or other graphical program

- **B.** Where can I find it?
  - **1.** Real TTY
  - **2.** Graphical Console
    - **a.** xterm
    - **b.** konsole
    - **c.** gnome-terminal
    - **d.** konqueror file manager terminal pane
  - **3.** Graphical "Run-Command" box
    - **a.** <Ctrl><F2> in both KDE and Gnome.
    - **b.** Some menus and tool bars provide one.
    - **c.** Interactions and viewing of error messages can be problematic.

- **C.** What does it do?
  - **1.** Reads and acts upon keyboard input.
  - **2.** Can also interact with pointing device (mouse)
    - **a.** gpm
    - **b.** gpm like buffer in X environment
    - **c.** Window manager may also provide a clipboard.

- **D.** What can I do with it?
  - **1.** Interact with "system objects"
    - **a.** File system objects - just "files" to experienced *nix users.
      - 1) Extends beyond what new users typically consider "files"
      - 2) User files
      - 3) Directories
      - 4) Symlinks
      - 5) Also device files, pipes and fifos
    - **b.** "Memory objects"
      - 1) Variables
      - 2) Aliases
      - 3) Functions
      - 4) Running processes
      - 5) System status flags

II. The command line interface- Why?

- **A.** GUI environment may not be available.
  - **1.** Graphic driver may fail.
  - **2.** Remote login

- **B.** Keeps hands on keyboard
  - **1.** Having to switch tools is always distracting.
  - **2.** When typing, try to keep using keyboard.
  - **3.** When mousing, keep using mouse.
  - **4.** Switch as infrequently as possible.

- **C.** Shell uses minimal system resources.
  - **1.** CPU -almost none
  - **2.** Memory very little
- **D.** Gateway to simple programming
  - **1.** Shell syntax is a programming language.
  - **2.** Special commands and keywords only make sense in program context.

- **E.** Great capabilities (power)
  - **1.** Every GUI gesture *could* be represented by a command string
  - **2.** Not every command string has a corresponding GUI gesture
  - **3.** Example of "entirely GUI" word processor.
    - **a.** Drop down menu for each part of speech: noun, verb, adjective, etc.
    - **b.** Obviously cumbersome.
    - **c.** Invariably limited vocabulary
    - **d.** This is why real word processors take keyboard input.
  - **4.** Concise, command line as simple or elaborate as it needs to be.

III. The command line interface – How?

- **A.** Set variables
- **B.** Execute commands.
- **C.** Source files.

IV. The command line interface- Your console environment

- **A.** Standard input = console
  - **1.** keyboard usage
    - **a.** Simple typing of command strings
    - **b.** Command history by cursor keys <Up/Down>
    - **c.** Command completion with <Tab> key
    - **d.** Home, End, & Ctrl key jumping
    - **e.** <ctrl>+<Left/Right> "word jumping
    - **f.** Carrots (^x^y^) fix typos
    - **g.** Ctrl- key combinations
      - 1) <Ctrl>+C Stop current process- kill current line
      - 2) <Ctrl>+U Erase current line
      - 3) <Ctrl>+D End of input (logout if at command prompt)
      - 4) <Ctrl>+Z Suspend input

5) <Ctrl>+V Next key sequence literal
6) <Ctrl>+L Clear or refresh screen

h. Scroll back of display <Shift>+<PgUp> <Shift>+<PgDn>

2. Mouse input to console

B. Standard output
1. Normally console
2. Can be redirected

C. Standard error =console
1. Normally console
2. Can be redirected

V. The command line interface – Basic Commands

A. *Typical* (linux) command syntax: `command` [-switches] [source] [target]
1. reason to avoid using '-' as first character in file name
2. Some commands use '-' to indicate single character arguments and '--' for whole words
3. end of switches often indicated by '--'
3. T*ypical* and *often* are cautionary: see documentation for command specific switch handling

B. Commands to tell you about other commands -Accessing documentation
1. apropos [keyword] or man -k [keyword]
2. type and which [command]
3. man and info ('q' quits either)

C. Commands that tell you something and then exit
1. pwd [-P]
2. echo
3. cat
4. more & less [-S -X] (q quits)
5. head & tail [-n *N*]
6. "list" commands
a. ls [-l/o -a/A -d -i]
b. ps [-A]
c. lspci [-v -vv]
d. lsmod
e. lsof
f. lsscsi *****
g. lscpu *****
h. lspath *****

D. Commands to filter what other commands "say" (receive their input from pipe "|")
1. grep [-i -r -v]
2. sort [-f -k -n *N* -r]
3. head & tail [-n -f (<Ctrl+C> quits tail -f)]
4. more & less are also filters
5. cut [-d -f]
6. uniq [-d -u -i]
7. od [-A n -t x*N*]
8. sed & awk "languages in themselves."

E. Quick examples

1. lscpu
2. lspath

F. More commands that tell tyou something and then exit

1. last [-i -n *N*]
2. who
3. whoami
4. whois
5. dig
6. netstat [-a -t -n]
7. ifconfig [interface]
8. traceroute
9. demesg

G. **C**ommands for writing text to files

1. Output redirection with '>'
2. Text editors
  a. vi (vim, gvim)
  b. joe
  c. pico
  d. kedit, etc

H. Commands for manipulating your "environment"

1. String delaration
  a. Not to be confused with "set' command (other shell environments)
  b. Can be used "self referentially"
  c. Simple integer math (on all numeral strings)
2. Set command

I. Commands used to run other commands

1. alias
2. function
3. source
4. exec

J. Commands for manipulating filesystem objects

1. cd [-]
2. mkdir [-p]
3. chmod [-a +/-rwxs] (or numeric)
4. chown [-r] *username*[:]
5. touch