



Intelligent
Systems
Lab

Administration of Operating Systems

GNU/Linux Utilities

Chapter 5

November 3, 2011

Key concepts



- Command line interface
- Commands, arguments, options
- Command line editing, history and completion
- Input/output, redirection, pipes
- Example utilities
- Powerful editors

Command Line Interface



- CLI is a mechanism for interacting with a computer by typing commands to perform specific tasks
 - text only
 - line-by-line input and output
 - less resources than GUI
- More difficult for beginners and occasional users
 - less feedback
 - no clearly visible presentation of options
- More flexible for expert users
 - allows tighter control over environment
 - unmatched automation capabilities

Commands



- GNU/Linux shell reads command one at a time
 - end with `<ENTER>`
- Everything until the first whitespace is a command name
 - internal command
 - program to execute
- Afterwards come command arguments
 - also separated by whitespaces
- Some characters have special meaning
 - `& ; | * ? ' " \ $! ~`
 - `[] () < > { } # / \`

Command Line Basics



- Each command starts at command prompt
 - `slawek@flora:~$`
 - command prompt indicates that shell is ready for next command
 - and lack of it means it is not
- Specify command name, options and arguments
 - options and arguments depend on a command
 - can, but don't have to, be optional
- GNU command line guidelines
 - short options `ls -a`
 - long options `ls --all`
 - `--help` and `--version`

Too easy?



- Don't use tab-completion
 - command names, filenames, ...
- Don't use arrow-key history
 - up/down for previous/next command
- Don't use line editing
 - left/right, delete, backspace
- Don't use line navigation
 - ctrl-a for beginning of line
 - ctrl-e for end of line
 - ctrl-u for clearing the line
 - ctrl-r (and ctrl-s) for incremental search of command history

Shell



- In GNU/Linux, shell is a program that provides an interface between users and operating system
- There exist a lot of different shells
 - `sh`, `bash`, `csch`, `tcsh`
 - `zoidberg`, `fish`, `scsh`, ...
- We will be using `bash`
 - default shell in Ubuntu distribution
 - but users can switch shells at any time
- Accessing shell
 - virtual console (Ubuntu Server)
 - terminal emulator (from GUI)
 - remotely using `ssh`

Quoting



- If special characters are to be used literally, they need to be quoted
 - so shell knows not to treat them as usual
 - `ls my file`
- Backslash
 - `ls my\ file`
 - use `\\` for literal backslash
- Quotation marks
 - `ls 'my file'`
 - `ls "my file"`

Getting Information



- man
 - man man
 - “man - an interface to the on-line reference manuals”
 - press <q> to exit
- help
 - help on internal shell commands
 - help cd
- --help
 - for most commands
 - ls --help
- Google

- By default, current working directory is also shown in command prompt

```

slawek@flora:~$ ln -s StarTrek/ TV
slawek@flora:~$ ls -l
drwxr-xr-x 3 slawek slawek 4096 Nov  2 20:55 StarTrek
drwxr-xr-x 3 slawek slawek 4096 Oct 30 16:39 StarWars
lrwxrwxrwx 1 slawek slawek   9 Nov  2 20:58 TV -> StarTrek/
slawek@flora:~$ cd TV
slawek@flora:~/TV$ pwd
/home/slawek/TV
slawek@flora:~/TV$ pwd -P
/home/slawek/StarTrek

```

cd



- `cd ~`
- `cd`
- `cd StarTrek`
- `cd /home/slawek/StarTrek`
- `cd ..`
- `cd -`
- `CDPATH`

ls



- ls
- ls StarTrek
- ls *.txt
- ls -l
- ls -a
- sorting
 - ls -S
 - ls -t
 - ls -r



file

- Display information about file type
 - based on its contents

```
slawek@flora:~/TV$ file imdb.html
imdb.html: HTML document text
slawek@flora:~/TV$ file img.jpg
img.jpg: JPEG image data, JFIF standard 1.01
slawek@flora:~/TV$ file /etc/passwd
/etc/passwd: ASCII text
slawek@flora:~/TV$ file /bin/echo
/bin/echo: ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), dynamically linked (uses
shared libs), for GNU/Linux 2.6.18, stripped
```



- Remove file
- Safety option
 - `-i`
 - ask for confirmation
 - especially useful when removing many different files at the same time
 - `rm -i *`
- `rm -r`
- `rm -rf StartTrek/`
 - Caution: `rm -rf StartTrek /`

cp



- Copy files/directories
- `cp source destination`
- `-i`
 - interactive
 - ask before overwriting files
- `-r`
 - recursive
- `-l`
 - link
- `-p`
 - preserve (mode, ownership & timestamps)



- Move/rename files/directories
- `mv source destination`
- `-i`
 - interactive
 - ask before overwriting files
- `-u`
 - update
- `-b`
 - backup

echo



- Write arguments to the standard output
- Can be used to create files
 - `echo Hello > hello.txt`
- Useful with shell filename matching
 - `echo *`
 - display all files in current directory
- Environment variables
 - `echo $USER`
 - display current user

cat



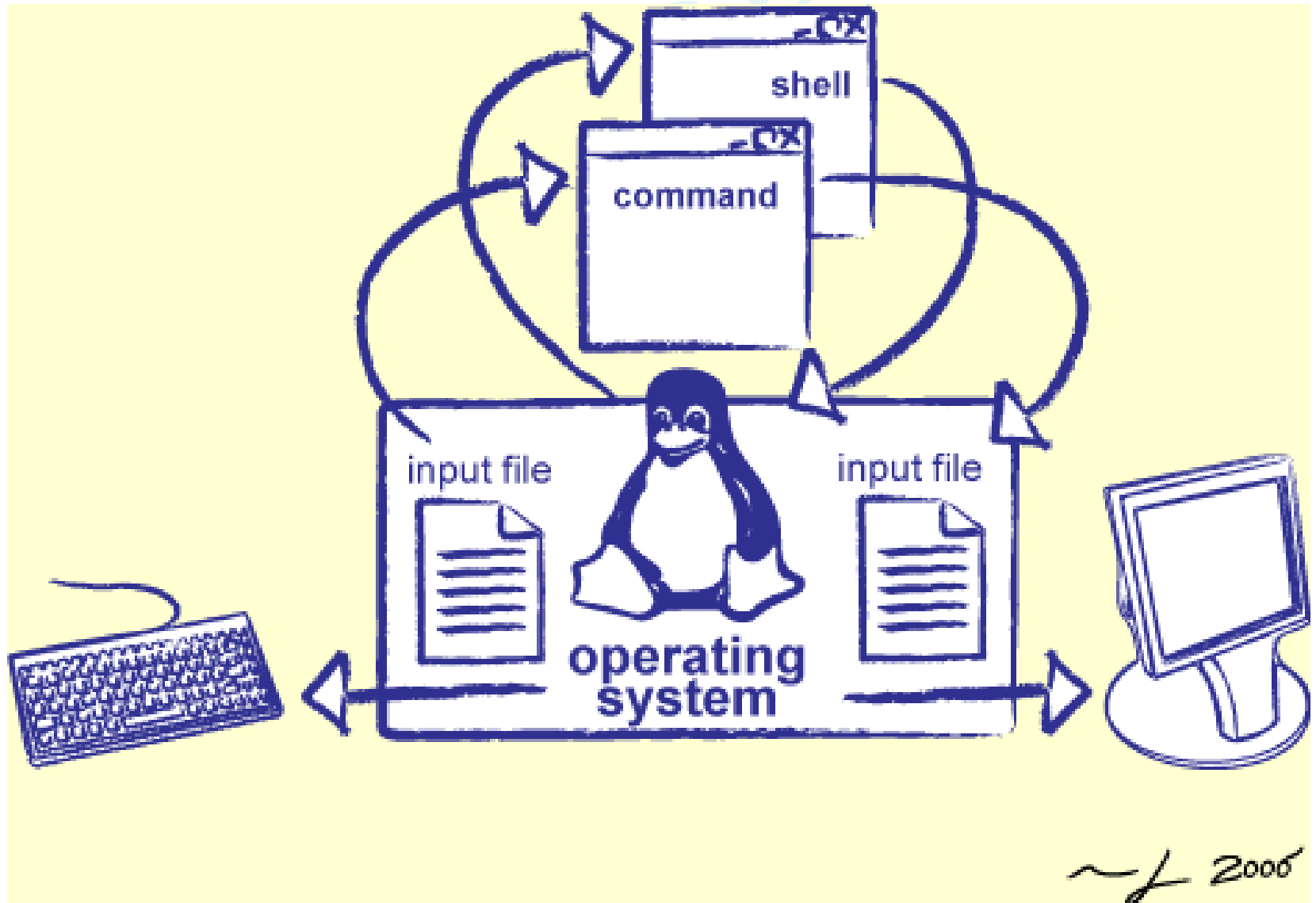
- “cat - concatenate files and print on the standard output”
- cat
- Most commonly used to display file contents
 - `cat hello.txt`
 - `cat < hello.txt`
- Can be used to “type” into a file
 - `cat > hello.txt`
- And finally, it can actually be used to concatenate several files
 - `cat file1 file2 > file3`

Standard Input & Output



- Each process in Linux has input and output channels between itself and its environment
 - standard input
 - standard output
 - standard error
- Behaviour of those channels follows a *file-like* protocol
- By default, input is connected to keyboard
 - and output is connected to the screen
- But this can be changed
 - very easily

Standard Input & Output



Input & Output Redirection



- `echo Hi > hello.txt`
 - special character `>` redirects standard output
 - instead of displaying “Hi” on the screen
 - echo will “display” it in a `hello.txt` file
- `cat < hello.txt`
 - special character `<` redirects standard input
 - instead reading data from the keyboard
 - notice that in this case we are calling `cat` without any arguments!
 - `cat` will read data from `hello.txt` file
- Both `<` and `>` are handled by the shell
 - utilities don’t really know (nor care) about it

less



- man less
 - “less - opposite of more”
- man more
 - “more - file perusal filter for crt viewing”
- less long_file
- Display file contents page-by-page
 - <SPACE> scroll one page
 - <ENTER> scroll one line
 - <q> quit
 - arrow keys, page up/down, ...

grep



- “grep, egrep, fgrep, rgrep - print lines matching a pattern”
- “grep searches the named input FILEs (or standard input if no files are named, or if a single hyphen-minus (-) is given as file name) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.”
- `grep ferengi TNG/episodes.html`
- `grep -i ferengi TNG/episodes.html`
- `grep -w`
- `grep -x`
- `grep -v`
- regular expressions: `grep -E`



- “wc - print newline, word, and byte counts for each file”
- Count how many TNG episodes featured Ferengi
 - `grep Ferengi episodes > tmp`
 - `wc tmp`
 - `5 195 2193`
 - `rm tmp`
- Works, but there are some problems
 - what if file `tmp` exists?
 - what if the file gets really big?
 - what if we forget to delete the file?
 - ...

Pipes



Intelligent
Systems
Lab

- Very simple but powerful form of communication between processes
 - `grep Ferengi episodes | wc`
 - directly connect `grep` output to `wc` input
- Very powerful technique
 - allowing virtually infinite number of combinations
 - very flexible interactions between powerful but specialised utilities
 - incredibly productive tool for experts
- Again, this is done by shell
 - utilities get this “for free”

head

- Display the first ten lines of the file

```
-c, --bytes=[-]K
  print the first K bytes of each file; with the
  leading '-', print all but the last K bytes of
  each file
-n, --lines=[-]K
  print the first K lines instead of the first 10;
  with the leading '-', print all but the last K
  lines of each file
-q, --quiet, --silent
  never print headers giving file names
-v, --verbose
  always print headers giving file names
--help
  display this help and exit
--version
  output version information and exit
```

tail



- Display the last ten lines of the file
- `-f`, `-follow[=name|descriptor]`
 - “output appended data as the file grows;”
- `-retry`
 - “keep trying to open a file even when it is or becomes inaccessible”

uniq



- Display a file, skipping *adjacent* duplicate lines
 - does not change the original file
- If you want to remove all duplicates, you may want to sort the file first
- `-s`, `-skip-chars=N`
 - avoid comparing the first N characters
- `-w`, `-check-chars=N`
 - compare no more than N characters in lines

sort



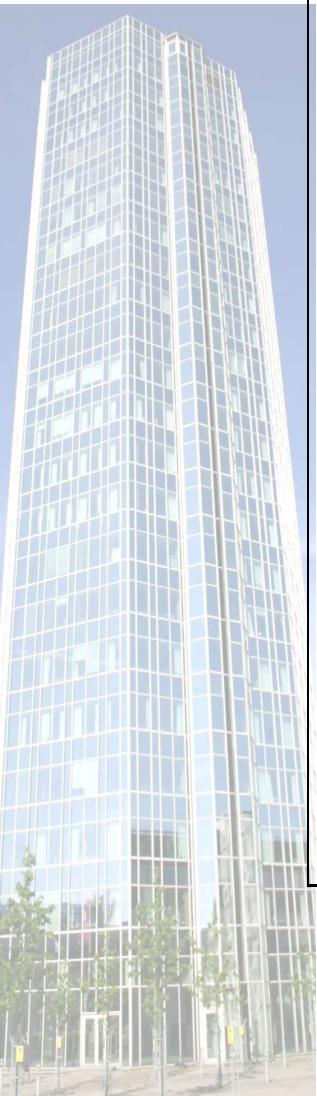
- `-b, -ignore-leading-blanks`
 - ignore leading blanks
- `-d, -dictionary-order`
 - consider only blanks and alphanumerics
- `-f, -ignore-case`
 - fold lower case to upper case characters
- `-M, -month-sort`
 - compare (unknown) < "JAN" < ... < "DEC"
- `-n, -numeric-sort`
 - compare according to string numerical value
- `-h, -human-numeric-sort`
 - compare human readable numbers: 2K<1G

diff



- Compare two files and display a list of differences between them
 - typically natural language or source code
- Algorithm is based on solving the longest common subsequence problem
- Breaks text into smaller *hunks*
- Three different output formats
 - traditional
 - context
 - unified
- Can be used as an input to patch
 - common in open source software development

diff



```
slawek@flora:~$ diff -u TNG/characters DS9/characters
--- TNG/characters      2011-11-02 22:20:09 +0100
+++ DS9/characters      2011-11-02 22:18:40 +0100
@@ -1,4 +1,4 @@
   Jean-Luc Picard
-William T. Riker
   Worf
-Data
+Sisko
+Odo
```

Finding files

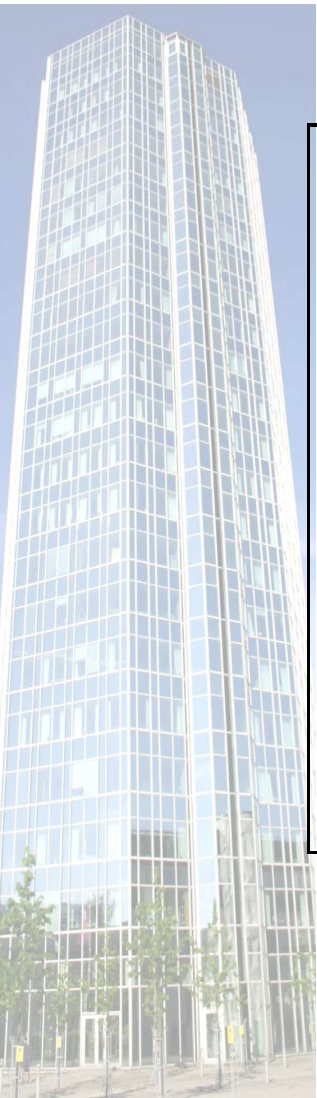


- which
 - search user's `$PATH`
 - find executable which will actually be called
- whereis
 - search standard locations
 - find binaries, source and documentation
- locate
 - search in a pre-generated database
- find
 - directly search on disk

cal

- Monthly calendar

```
slawek@flora:~$ cal
  November 2011
Su Mo Tu We Th Fr Sa
    1  2  3  4  5
  6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```



lpr



- *Line printer* utility
 - places one or more files in a print queue
 - for the default printer
- `lpr -p` allows to print on a specific printer
- `lpstat` displays information about printing jobs submitted by the user
 - `lpstat -p` shows available printers
 - `lpstat -o` displays all printing jobs

Compressing and Archiving



- bzip2
- bunzip2
- bzip2
- bzip2
- bzip2
- bzip2
- bzip2
- bzip2



Other Interesting Commands



- who, w, finger
- cut, tr
- date
- hostname
- script
- todos, fromdos
- mail
- du, df

Alias



- An alias is a name that the shell translates into another command
 - creating a nickname for complex command
- `alias duSub='du -k | grep -v "/.*/" | sort -n'`
- Can also be used to change default behaviour of some commands
 - `alias ls=ls -l`
 - `alias rm=rm -i`
 - `alias dir=ls -l`
- `alias` without arguments displays all currently defined aliases

Vim



- a *modal* editor
 - insert mode and command mode
- Entering insert mode
 - `<i>` insert, `<a>` append, `<r>/<R>` replace, ...
- `<ESCAPE>` exits insert mode
- `<:>` enters *last line mode*
 - command line interface
- Exit
 - without saving changes: `:q!`
 - with saving changes: `ZZ`
- Try the online tutorial
 - `vimtutor`

Emacs



Intelligent
Systems
Lab

- “the extensible, customizable, self-documenting, real-time display editor”
- Written (and programmable) in Lisp
 - with low-level part written in C
- One of the flagships of the GNU project
 - next to gcc
- Unparalleled customisability
 - due to the dynamic nature of Lisp
- Famous for multi-key shortcuts
 - C-x C-c to exit
- Huge amount of third-party libraries

Key concepts



- Command line interface
- Commands, arguments, options
- Command line editing, history and completion
- Input/output, redirection, pipes
- Example utilities
- Powerful editors



Intelligent
Systems
Lab

Questions?

