Purposes of this talk/workshop

- teach/remind you the basic Unix commands to be actually learn something from this 2 weeks courses
- avoid painfull mistakes
- give you good/healthy Unix habits
- show you some other commands that might be helpfull sometime in the future
- NOT teaching you system administration
Purposes of this talk/workshop

- teach/remind you the basic Unix commands to be actually learn something from this 2 weeks courses
- avoid painful mistakes
- give you good/healthy Unix habits
- show you some other commands that might be helpful sometime in the future
- NOT teaching you system administration
Purposes of this talk/workshop

- teach/remind you the basic Unix commands to be actually learn something from this 2 weeks courses
- avoid painfull mistakes
- give you good/healthy Unix habits
  - show you some other commands that might be helpfull sometime in the future
- NOT teaching you system administration
Purposes of this talk/workshop

- teach/remind you the basic Unix commands to be actually learn something from this 2 weeks courses
- avoid painfull mistakes
- give you good/healthy Unix habits
- show you some other commands that might be helpfull sometime in the future
- NOT teaching you system administration
Purposes of this talk/workshop

- teach/remind you the basic Unix commands to be actually learn something from this 2 weeks courses
- avoid painfull mistakes
- give you good/healthy Unix habits
- show you some other commands that might be helpfull sometime in the future
- NOT teaching you system administration
1998 Graduate student ESPCI, Chemist

2002 PhD in molecular modeling (folding mechanism of a multidomain protein), familiar with molecular dynamics simulation on small clusters to large computing center

2002 Join the Bioinformatic Group of P. M. Nilges at the Institut Pasteur as system admin (90 machines, 300 cores, 20 TB of shared storage/NFS, 20 users) http://www.pasteur.fr

2004 join the CentOS project http://www.centos.org
1998 Graduate student ESPCI, Chemist
2002 PhD in molecular modeling (folding mechanism of a multidomain protein), familiar with molecular dynamics simulation on small clusters to large computing center
2002 Join the Bioinformatic Group of P. M. Nilges at the Institut Pasteur as system admin (90 machines, 300 cores, 20 TB of shared storage/NFS, 20 users) http://www.pasteur.fr
2004 join the CentOS project http://www.centos.org
Who am I

- 1998 Graduate student ESPCI, Chemist
- 2002 PhD in molecular modeling (folding mechanism of a multidomain protein), familiar with molecular dynamics simulation on small clusters to large computing center
- 2002 Join the Bioinformatic Group of P. M. Nilges at the Institut Pasteur as system admin (90 machines, 300 cores, 20 TB of shared storage/NFS, 20 users) http://www.pasteur.fr
- 2004 join the CentOS project http://www.centos.org
Who am I

- 1998 Graduate student ESPCI, Chemist
- 2002 PhD in molecular modeling (folding mechanism of a multidomain protein), familiar with molecular dynamics simulation on small clusters to large computing center
- 2002 Join the Bioinformatic Group of P. M. Nilges at the Institut Pasteur as system admin (90 machines, 300 cores, 20 TB of shared storage/NFS, 20 users) http://www.pasteur.fr
- 2004 join the CentOS project http://www.centos.org
Basic level knowledge of unix systems.
User account informations

Account informations

- login
- password
- members of groups
- just 'numbers': uid, gid
- $HOME directory
User account informations

Account informations

- login
- password
- members of groups
- just 'numbers': uid, gid
- $HOME directory
User account informations

Account informations

- login
- password
- members of groups
  - just ‘numbers’: uid, gid
- $HOME directory
User account informations

Account informations

- login
- password
- members of groups
- just ’numbers’: uid, gid
- $HOME directory
User account informations

Account informations
- login
- password
- members of groups
- just ’numbers’: uid, gid
- $HOME directory
A multi users system: sharing resources

- processor resources
- storage space (disk: local or remote)
- superuser: root
A multi users system: sharing resources

- processor resources
- storage space (disk: local or remote)
- superuser: root
A multi users system: sharing resources

- processor resources
- storage space (disk: local or remote)
- superuser: root
files, folders or directories

```bash
dirA
 | - dirB
 |   | - dirC
 |   |   | - file1
 |   |   | - file2
 |   |   `- file3
 | `- dirD
    | - fileA
    `- fileB
```

**relative and absolute PATH:**

```
RELATIVE PATH:    ABSOLUTE PATH:
/                /
 | - etc
 | - home
 | - lib
 | - sbin
 | - tmp
 | - usr
 |   | - bin
 |   |   | - bash
 |   |   `- local
 |   |     | - bin
 |   |     | - lib
 |   `- var
 |     | - /usr/bin
 |     |   | - /usr/bin/bash
 |     |   `- /usr/local
 |     |     | - bin
 |     |     | - lib
 |     `- var
```
Command line interface

Interaction through the keyboard ONLY . . .

- the SHELL: interpret the words (space separated) you TYPE . . .
  - the case is IMPORTANT
    - `ls` `LS` `lS` `Ls`
    - are all different commands
  - which shell? bash (default shell on linux) tcsh or zsh

What is a command?

  username@servername > programname -option1 ... argument1 ...
  tru@callab > ls -l -d /home
Interaction through the keyboard ONLY . . .

- the SHELL: interpret the words (space separated) you TYPE . . .
- the case is IMPORTANT

```
ls LS lS Ls
```

are all different commands

- which shell? bash (default shell on linux) tcsh or zsh
- What is a command?

```
username@servername > programname -option1 ... argument1 ...
tru@callab > ls -l -d /home
```
Interaction through the keyboard ONLY . . .

- the SHELL: interpret the words (space separated) you TYPE . . .
- the case is IMPORTANT

`ls LS ls Ls`

are all different commands

- which shell? bash (default shell on linux) tcsh or zsh

What is a command?

`username@servername > programname -option1 ... argument1 ...`
`tru@callab > ls -l -d /home`
Interaction through the keyboard ONLY . . .

- the SHELL: interpret the words (space separated) you TYPE . . .

- the case is IMPORTANT

  ls LS lS Ls

  are all different commands

- which shell? bash (default shell on linux) tcsh or zsh

- What is a command?

  username@servername > programname -option1 ... argument1 ...
  tru@callab > ls -l -d /home
Graphical user interface

- historical: Xerox has invented the X windows and the mouse
  - Apple/Macintosh made the mouse popular (one button), drag/drop
  - Microsoft windows(TM) made it popular (cut/paste with ctrl-c/v) for all the others
  - Linux/Unix GUI: highlight with the left button pressed, paste text in X windows -> middle mouse
Graphical user interface

- historical: Xerox has invented the X windows and the mouse
- Apple/Macintosh made the mouse popular (one button), drag/drop
  - Microsoft windows(TM) made it popular (cut/paste with ctrl-c/v) for all the others
- Linux/Unix GUI: highlight with the left button pressed, paste text in X windows -> middle mouse
Graphical user interface

- historical: Xerox has invented the X windows and the mouse
- Apple/Macintosh made the mouse popular (one button), drag/drop
- Microsoft windows(TM) made it popular (cut/paste with ctrl-c/v) for all the others
- Linux/Unix GUI: highlight with the left button pressed, paste text in X windows -> middle mouse
Graphical user interface

- historical: Xerox has invented the X windows and the mouse
- Apple/Macintosh made the mouse popular (one button), drag/drop
- Microsoft windows(TM) made it popular (cut/paste with ctrl-c/v) for all the others
- Linux/Unix GUI: highligth with the left button pressed, paste text in X windows -> middle mouse
Operating systems overview

- **PC under Windows**
  - Commercial Unix (proprietary hardware): AIX, HPUX, Solaris, and dead now: IRIX(SGI), Tru64(Dec/Compaq/HP)
  - Opensource Unix (MIT license/commodity hardware): *BSD, OpenSolaris
  - Opensource Unix (GPL/commodity hardware): Linux (free or commercially supported)
  - MacOSX (proprietary hardware with based on commodity hardware)
  - but now standard to intel compatible
Operating systems overview

- PC under Windows
- Commercial Unix (proprietary hardware): AIX, HPUX, Solaris, and dead now: IRIX(SGI), Tru64(Dec/Compaq/HP)
- Opensource Unix (MIT license/commodity hardware): *BSD, OpenSolaris
- Opensource Unix (GPL/commodity hardware): Linux (free or commercially supported)
- MacOSX (proprietary hardware with based on commodity hardware)
- but now standard to intel compatible
PC under Windows

Commercial Unix (proprietary hardware): AIX, HPUX, Solaris, and dead now: IRIX(SGI), Tru64(Dec/Compaq/HP)

Opensource Unix (MIT license/commodity hardware): *BSD, OpenSolaris

Opensource Unix (GPL/commodity hardware): Linux (free or commercially supported)

MacOSX (proprietary hardware with based on commodity hardware)

but now standard to intel compatible
Operating systems overview

- PC under Windows
- Commercial Unix (proprietary hardware): AIX, HPUX, Solaris, and dead now: IRIX(SGI), Tru64(Dec/Compaq/HP)
- Opensource Unix (MIT license/commodity hardware): *BSD, OpenSolaris
- Opensource Unix (GPL/commodity hardware): Linux (free or commercially supported)
- MacOSX (proprietary hardware with based on commodity hardware)
- but now standard to intel compatible
Operating systems overview

- PC under Windows
- Commercial Unix (proprietary hardware): AIX, HPUX, Solaris, and dead now: IRIX(SGI), Tru64(Dec/Compaq/HP)
- Opensource Unix (MIT license/commodity hardware): *BSD, OpenSolaris
- Opensource Unix (GPL/commodity hardware): Linux (free or commercially supported)
- MacOSX (proprietary hardware with based on commodity hardware)

but now standard to intel compatible
Operating systems overview

- PC under Windows
- Commercial Unix (proprietary hardware): AIX, HPUX, Solaris, and dead now: IRIX(SGI), Tru64(Dec/Compaq/HP)
- Opensource Unix (MIT license/commodity hardware): *BSD, OpenSolaris
- Opensource Unix (GPL/commodity hardware): Linux (free or commercially supported)
- MacOSX (proprietary hardware with based on commodity hardware)
- but now standard to intel compatible
Virtualisation and multiboot

What next?

- multiple boot the same hardware
- virtualisation (xen, vmware, virtualbox, ...)
- jail/zones/chroot
- same hardware/different OS/same time or not
- different hardware, but same OS: see NetBSD for example
  - http://www.netbsd.org/Ports/: 57 supported ports
    - standard PC 32/64 bits
    - sparc
    - alpha, mips, powerpc
    - ...
- multi arches software (firefox, Acrobat Reader, OpenOffice, skype, ... python, perl, ...)

[Image of the left sidebar with menu options such as Working with linux, Basic level, Multiple OS, Remote connection, First commands, Regular expressions, Moving around in the filesystem, Rights and permissions, Organise your project, Self help]
Virtualisation and multiboot

What next?

- multiple boot the same hardware
- virtualisation (xen, vmware, virtualbox, . . . )
  jail/zones/chroot
- same hardware/different OS/same time or not
- different hardware, but same OS: see NetBSD for example
  http://www.netbsd.org/Ports/: 57 supported ports
  - standard PC 32/64 bits
  - sparc
  - alpha, mips, powerpc
  - . . .

- multi arches software (firefox, Acrobat Reader, OpenOffice, skype, . . . python, perl, . . .)
Virtualisation and multiboot

What next?

- multiple boot the same hardware
- virtualisation (xen, vmware, virtualbox, ...)
  jail/zones/chroot
- same hardware/different OS/same time or not
- different hardware, but same OS: see NetBSD for example
  http://www.netbsd.org/Ports/: 57 supported ports
  - standard PC 32/64 bits
  - sparc
  - alpha, mips, powerpc
  - ...
- multi arches software (firefox, Acrobat Reader, OpenOffice, skype, ... python, perl,..)
Virtualisation and multiboot

What next?

- multiple boot the same hardware
- virtualisation (xen, vmware, virtualbox, ... ) jail/zones/chroot
- same hardware/different OS/same time or not
- different hardware, but same OS: see NetBSD for example
  http://www.netbsd.org/Ports/: 57 supported ports
  - standard PC 32/64 bits
  - sparc
  - alpha, mips, powerpc
  - ...

- multi arches software (firefox, Acrobat Reader, OpenOffice, skype, ... python, perl,..)
Virtualisation and multiboot

What next?

- multiple boot the same hardware
- virtualisation (xen, vmware, virtualbox, ...)
  jail/zones/chroot
- same hardware/different OS/same time or not
- different hardware, but same OS: see NetBSD for example
  http://www.netbsd.org/Ports/: 57 supported ports
  - standard PC 32/64 bits
    - sparc
    - alpha, mips, powerpc
    - ...

- multi arches software (firefox, Acrobat Reader, OpenOffice, skype, ... python, perl, ...
Virtualisation and multiboot

What next?

- multiple boot the same hardware
- virtualisation (xen, vmware, virtualbox, ... ) jail/zones/chroot
- same hardware/different OS/same time or not
- different hardware, but same OS: see NetBSD for example
  http://www.netbsd.org/Ports/: 57 supported ports
  - standard PC 32/64 bits
  - sparcs
    - alpha, mips, powerpc
    - ... 
  - multi arches software (firefox, Acrobat Reader, OpenOffice, skype, ... python, perl, ... )
Virtualisation and multiboot

What next?

- multiple boot the same hardware
- virtualisation (xen, vmware, virtualbox, ...)
  jail/zones/chroot
- same hardware/different OS/same time or not
- different hardware, but same OS: see NetBSD for example
  http://www.netbsd.org/Ports/: 57 supported ports
  - standard PC 32/64 bits
  - sparc
  - alpha, mips, powerpc
  - ...

- multi arches software (firefox, Acrobat Reader, OpenOffice, skype, ... python, perl,..)
Virtualisation and multiboot

What next?

- multiple boot the same hardware
- virtualisation (xen, vmware, virtualbox, ... )
  jail/zones/chroot
- same hardware/different OS/same time or not
- different hardware, but same OS: see NetBSD for example
  http://www.netbsd.org/Ports/
  57 supported ports
  - standard PC 32/64 bits
  - sparc
  - alpha, mips, powerpc
  - ...

- multi arches software (firefox, Acrobat Reader, OpenOffice, skype, ... python, perl,..)
Virtualisation and multiboot

What next?

- multiple boot the same hardware
- virtualisation (xen, vmware, virtualbox, ...) jail/zones/chroot
- same hardware/different OS/same time or not
- different hardware, but same OS: see NetBSD for example
  http://www.netbsd.org/Ports/: 57 supported ports
    - standard PC 32/64 bits
    - sparc
    - alpha, mips, powerpc
    - ...
- multi arches software (firefox, Acrobat Reader, OpenOffice, skype, ... python, perl, ..)
Connecting to another machine (download/upload/remote shell)

- N to 1: client(s) to 1 server
  - different protocols
    - ftp:// (use ftp or better wget and lftp)
    - http:// (regular browser, wget, elinks lftp)
    - ssh
  - any N to any M: peer to peer (torrent)
Connecting to another machine (download/upload/remote shell)

- N to 1: client(s) to 1 server
- different protocols
  - ftp:// (use ftp or better wget and lftp)
  - http:// (regular browser, wget, elinks lftp)
  - ssh
- any N to any M: peer to peer (torrent)
Connecting to another machine (download/upload/remote shell)

- N to 1: client(s) to 1 server
- different protocols
  - ftp:// (use ftp or better wget and lftp)
  - http:// (regular browser, wget, elinks lftp)
  - ssh
- any N to any M: peer to peer (torrent)
Connecting to another machine (download/upload/remote shell)

- N to 1: client(s) to 1 server
- different protocols
  - ftp:// (use ftp or better wget and lftp)
  - http:// (regular browser, wget, elinks lftp)
  - ssh
- any N to any M: peer to peer (torrent)
Connecting to another machine (download/upload/remote shell)

- N to 1: client(s) to 1 server
- different protocols
  - ftp:// (use ftp or better wget and lftp)
  - http:// (regular browser, wget, elinks lftp)
  - ssh
- any N to any M: peer to peer (torrent)
Connecting to another machine (download/upload/remote shell)

- N to 1: client(s) to 1 server
- different protocols
  - ftp:// (use ftp or better wget and lftp)
  - http:// (regular browser, wget, elinks lftp)
  - ssh
- any N to any M: peer to peer (torrent)
My 1st commands

- **whoami**: display the user name
- **id**: display the user identity (login, uid, group(s), gid)
- **date**: display the date, timezone
- **hostname**: display the name of the server
- **who, w, uptime**: who is logged? what they are doing, how long the system has been running?
- **passwd**: change your password
- **top**: interactive display sorted processes running
- **firefox**: your usual web browser client
- **ctrl-c**: tell current the program to stop
- **&, ctrl-z, bg, fg**: job control
- **alias**: display/create command lines shortcuts
My 1st commands

- **whoami**: display the user name
- **id**: display the user identity (login, uid, group(s), gid)
- **date**: display the date, timezone
- **hostname**: display the name of the server
- **who, w, uptime**: who is logged? what they are doing, how long the system has been running?
- **passwd**: change your password
- **top**: interactive display sorted processes running
- **firefox**: your usual web browser client
- **ctrl-c**: tell current the program to stop
- **&, ctrl-z, bg, fg**: job control
- **alias**: display/create command lines shortcuts
My 1st commands

- **whoami**: display the user name
- **id**: display the user identity (login, uid, group(s), gid)
- **date**: display the date, timezone
- **hostname**: display the name of the server
- **who, w, uptime**: who is logged? what they are doing, how long the system has been running?
- **passwd**: change your password
- **top**: interactive display sorted processes running
- **firefox**: your usual web browser client
- **ctrl-c**: tell current the program to stop
- **&**, **ctrl-z**, **bg**, **fg**: job control
- **alias**: display/create command lines shortcuts
My 1st commands

- `whoami`: display the user name
- `id`: display the user identity (login, uid, group(s), gid)
- `date`: display the date, timezone
- `hostname`: display the name of the server
- `who`, `w`, `uptime`: who is logged? what they are doing, how long the system has been running?
- `passwd`: change your password
- `top`: interactive display sorted processes running
- `firefox`: your usual web browser client
- `ctrl-c`: tell current the program to stop
- `&`, `ctrl-z`, `bg`, `fg`: job control
- `alias`: display/create command lines shortcuts
My 1st commands

- `whoami`: display the user name
- `id`: display the user identity (login, uid, group(s), gid)
- `date`: display the date, timezone
- `hostname`: display the name of the server
- `who`, `w`, `uptime`: who is logged? what they are doing, how long the system has been running?
- `passwd`: change your password
- `top`: interactive display sorted processes running
- `firefox`: your usual web browser client
- `ctrl-c`: tell current the program to stop
- `&`, `ctrl-z`, `bg`, `fg`: job control
- `alias`: display/create command lines shortcuts
My 1st commands

- **whoami**: display the user name
- **id**: display the user identity (login, uid, group(s), gid)
- **date**: display the date, timezone
- **hostname**: display the name of the server
- **who, w, uptime**: who is logged? what they are doing, how long the system has been running?
- **passwd**: change your password
- **top**: interactive display sorted processes running
- **firefox**: your usual web browser client
- **ctrl-c**: tell current the program to stop
- **&, ctrl-z, bg, fg**: job control
- **alias**: display/create command lines shortcuts
My 1st commands

- `whoami`: display the user name
- `id`: display the user identity (login, uid, group(s), gid)
- `date`: display the date, timezone
- `hostname`: display the name of the server
- `who`, `w`, `uptime`: who is logged? what they are doing, how long the system has been running?
- `passwd`: change your password
- `top`: interactive display sorted processes running
- `firefox`: your usual web browser client
- `ctrl-c`: tell current the program to stop
- `&`, `ctrl-z`, `bg`, `fg`: job control
- `alias`: display/create command lines shortcuts
My 1st commands

- `whoami`: display the user name
- `id`: display the user identity (login, uid, group(s), gid)
- `date`: display the date, timezone
- `hostname`: display the name of the server
- `who, w, uptime`: who is logged? what they are doing, how long the system has been running?
- `passwd`: change your password
- `top`: interactive display sorted processes running
- `firefox`: your usual web browser client
- `ctrl-c`: tell current the program to stop
- `&`, `ctrl-z`, `bg`, `fg`: job control
- `alias`: display/create command lines shortcuts
My 1st commands

- `whoami`: display the user name
- `id`: display the user identity (login, uid, group(s), gid)
- `date`: display the date, timezone
- `hostname`: display the name of the server
- `who`, `w`, `uptime`: who is logged? what they are doing, how long the system has been running?
- `passwd`: change your password
- `top`: interactive display sorted processes running
- `firefox`: your usual web browser client
- `ctrl-c`: tell current the program to stop
- `&`, `ctrl-z`, `bg`, `fg`: job control
- `alias`: display/create command lines shortcuts
My 1st commands

- whoami: display the user name
- id: display the user identity (login, uid, group(s), gid)
- date: display the date, timezone
- hostname: display the name of the server
- who, w, uptime: who is logged? what they are doing, how long the system has been running?
- passwd: change your password
- top: interactive display sorted processes running
- firefox: your usual web browser client
- ctrl-c: tell current the program to stop
- &, ctrl-z, bg, fg: job control
- alias: display/create command lines shortcuts
**My 1st commands**

- `whoami`: display the user name
- `id`: display the user identity (login, uid, group(s), gid)
- `date`: display the date, timezone
- `hostname`: display the name of the server
- `who, w, uptime`: who is logged? what they are doing, how long the system has been running?
- `passwd`: change your password
- `top`: interactive display sorted processes running
- `firefox`: your usual web browser client
- `ctrl-c`: tell current the program to stop
- `&`, `ctrl-z`, `bg`, `fg`: job control
- `alias`: display/create command lines shortcuts
Regular expressions: REGEX

- A[abz]: A and one character (a or b or z)
- A[1-9]: A and one digit from 1 to 9
- *: a chain of any length with any character
- *.txt: a chain of any length with any character terminated by .txt
- *[Aa]*.txt: a chain of any length with any character containing an upper/lower case A and terminated by .txt
Regular expressions: REGEX

- A[abz]: A and one character (a or b or z)
- A[1-9]: A and one digit from 1 to 9
- * : a chain of any length with any character
- *.txt : a chain of any length with any character terminated by .txt
- *[Aa]*.txt : a chain of any length with any character containing an upper/lower case A and terminated by .txt
Regular expressions: REGEX

- A[abz]: A and one character (a or b or z)
- A[1-9]: A and one digit from 1 to 9
- *: a chain of any length with any character
- *.txt: a chain of any length with any character terminated by .txt
- *[Aa]*.txt: a chain of any length with any character containing an upper/lower case A and terminated by .txt
Regular expressions: REGEX

- A[abz]: A and one character (a or b or z)
- A[1-9]: A and one digit from 1 to 9
- *: a chain of any length with any character
- *.txt: a chain of any length with any character terminated by .txt
- *[Aa]*.txt: a chain of any length with any character containing an upper/lower case A and terminated by .txt
Regular expressions: REGEX

- A[abz]: A and one character (a or b or z)
- A[1-9]: A and one digit from 1 to 9
- * : a chain of any length with any character
- *.txt : a chain of any length with any character terminated by .txt
- *[Aa]*.txt : a chain of any length containing an upper/lower case A and terminated by .txt
Files and directories

- **pwd**: print working directory
- `. and ..`: the names of the current directory/the one above
- **cd**: change directory (move around the filesystem)
- **ls** (and options): list files/directories
- **file**: display the file type
- **mv**: move, rename files/directories
- **cp**: copy files/directories
- **rm**: remove files
- **mkdir**, **rmdir**: create/remove directories
- **cat**, **echo**: display the content of a file/variable
- **more**, **less**: pagers (type q to quit)
- **cmp**, **diff**: compare files
Files and directories

- **pwd**: print working directory
- `. and ..`: the names of the current directory/the one above
- **cd**: change directory (move around the filesystem)
- **ls** (and options): list files/directories
- **file**: display the file type
- **mv**: move, rename files/directories
- **cp**: copy files/directories
- **rm**: remove files
- **mkdir, rmdir**: create/remove directories
- **cat, echo**: display the content of a file/variable
- **more, less**: pagers (type q to quit)
- **cmp, diff**: compare files
Files and directories

- `pwd`: print working directory
- `. and ..`: the names of the current directory/the one above
- `cd`: change directory (move around the filesystem)
  - `ls` (and options): list files/directories
  - `file`: display the file type
  - `mv`: move, rename files/directories
  - `cp`: copy files/directories
  - `rm`: remove files
  - `mkdir, rmdir`: create/remove directories
  - `cat, echo`: display the content of a file/variable
  - `more, less`: pagers (type q to quit)
  - `cmp, diff`: compare files
Files and directories

- `pwd`: print working directory
- `.` and `..`: the names of the current directory/the one above
- `cd`: change directory (move around the filesystem)
- `ls` (and options): list files/directories
  - `file`: display the file type
  - `mv`: move, rename files/directories
  - `cp`: copy files/directories
  - `rm`: remove files
  - `mkdir`, `rmdir`: create/remove directories
  - `cat`, `echo`: display the content of a file/variable
  - `more`, `less`: pagers (type q to quit)
  - `cmp`, `diff`: compare files
Files and directories

- `pwd`: print working directory
- `.` and `..`: the names of the current directory/the one above
- `cd`: change directory (move around the filesystem)
- `ls` (and options): list files/directories
- `file`: display the file type
- `mv`: move, rename files/directories
- `cp`: copy files/directories
- `rm`: remove files
- `mkdir`, `rmdir`: create/remove directories
- `cat`, `echo`: display the content of a file/variable
- `more`, `less`: pagers (type q to quit)
- `cmp`, `diff`: compare files
Files and directories

- **pwd**: print working directory
- . and .. : the names of the current directory/the one above
- **cd**: change directory (move around the filesystem)
- **ls** (and options): list files/directories
- **file**: display the file type
- **mv**: move, rename files/directories
- **cp**: copy files/directories
- **rm**: remove files
- **mkdir, rmdir**: create/remove directories
- **cat, echo**: display the content of a file/variable
- more, less: pagers (type q to quit)
- **cmp, diff**: compare files
Files and directories

- `pwd`: print working directory
- `. and ..`: the names of the current directory/the one above
- `cd`: change directory (move around the filesystem)
- `ls` (and options): list files/directories
- `file`: display the file type
- `mv`: move, rename files/directories
- `cp`: copy files/directories
- `rm`: remove files
- `mkdir, rmdir`: create/remove directories
- `cat, echo`: display the content of a file/variable
- `more, less`: pagers (type q to quit)
- `cmp, diff`: compare files
Files and directories

- `pwd`: print working directory
- `. and ..`: the names of the current directory/the one above
- `cd`: change directory (move around the filesystem)
- `ls` (and options): list files/directories
- `file`: display the file type
- `mv`: move, rename files/directories
- `cp`: copy files/directories
- `rm`: remove files
- `mkdir, rmdir`: create/remove directories
- `cat, echo`: display the content of a file/variable
- `more, less`: pagers (type q to quit)
- `cmp, diff`: compare files
Files and directories

- **pwd**: print working directory
- `.` and `..`: the names of the current directory/the one above
- **cd**: change directory (move around the filesystem)
- **ls** (and options): list files/directories
- **file**: display the file type
- **mv**: move, rename files/directories
- **cp**: copy files/directories
- **rm**: remove files
- **mkdir, rmdir**: create/remove directories
- **cat, echo**: display the content of a file/variable
- **more, less**: pagers (type `q` to quit)
- **cmp, diff**: compare files
Files and directories

- **pwd**: print working directory
- `. and ..`: the names of the current directory/the one above
- **cd**: change directory (move around the filesystem)
- **ls** (and options): list files/directories
- **file**: display the file type
- **mv**: move, rename files/directories
- **cp**: copy files/directories
- **rm**: remove files
- **mkdir, rmdir**: create/remove directories
- **cat, echo**: display the content of a file/variable
- **more, less**: pagers (type q to quit)
- **cmp, diff**: compare files
Files and directories

- `pwd`: print working directory
- `..`: the names of the current directory/the one above
- `cd`: change directory (move around the filesystem)
- `ls` (and options): list files/directories
- `file`: display the file type
- `mv`: move, rename files/directories
- `cp`: copy files/directories
- `rm`: remove files
- `mkdir`, `rmdir`: create/remove directories
- `cat`, `echo`: display the content of a file/variable
- `more`, `less`: pagers (type q to quit)
- `cmp`, `diff`: compare files
Files and directories

- **pwd**: print working directory
- `.` and `..`: the names of the current directory/the one above
- **cd**: change directory (move around the filesystem)
- **ls** (and options): list files/directories
- **file**: display the file type
- **mv**: move, rename files/directories
- **cp**: copy files/directories
- **rm**: remove files
- **mkdir**, **rmdir**: create/remove directories
- **cat**, **echo**: display the content of a file/variable
- **more**, **less**: pagers (type q to quit)
- **cmp**, **diff**: compare files
Rights and permissions on files

-`ls -l`

- `umask`: display/set the default permissions
- `chown`: change the owner
- `chgrp`: change the group
- `chmod`: change the permission
## Rights and permissions on files

<table>
<thead>
<tr>
<th>Permissions</th>
<th>User</th>
<th>Group</th>
<th>Others</th>
<th>Date/Time</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>-rw--r--r--</td>
<td>1</td>
<td>tru</td>
<td>Bis</td>
<td>268 Nov 16 19:23</td>
<td>file1</td>
</tr>
<tr>
<td>drwxr-xr-x</td>
<td>2</td>
<td>tru</td>
<td>Bis</td>
<td>4096 Nov 16 19:23</td>
<td>dir2</td>
</tr>
</tbody>
</table>

uuugggoooo

<table>
<thead>
<tr>
<th>Rights/Perms</th>
<th>r=4, w=2, x=1</th>
</tr>
</thead>
</table>

- **ls -l**
- **umask**: display/set the default permissions
- **chown**: change the owner
- **chgrp**: change the group
- **chmod**: change the permission
Rights and permissions on files

- `ls -l`

- `umask`: display/set the default permissions

- `chown`: change the owner

- `chgrp`: change the group

- `chmod`: change the permission
Rights and permissions on files

-Is -l

- umask: display/set the default permissions
- chown: change the owner
- chgrp: change the group
- chmod: change the permission

```
-rw-r--r-- 1 tru Bis 268 Nov 16 19:23 file1
drwxr-xr-x 2 tru Bis 4096 Nov 16 19:23 dir2
```

```
-uuugggoo
  <-|  |->
  |   |
  |     ---------
  | rights/perms r=4, w=2, x=1
  ---------
```
Rights and permissions on files

ls -l

umask: display/set the default permissions

chown: change the owner

chgrp: change the group

chmod: change the permission
Managing resources

- **processor resources:** top, ps, kill, nice, renice
  - ps: display processes
  - top: continuously display sorted processes
  - kill: send signals to process (stop, suspend, continue)
  - nice, renice: lower the priority
  - ulimit: display imposed limits
  - free: display the memory usage

- **storage**
  - df: disk free
  - du: disk used
Managing resources

- **processor resources**: `top`, `ps`, `kill`, `nice`, `renice`
  - `ps`: display processes
  - `top`: continuously display sorted processes
  - `kill`: send signals to process (stop, suspend, continue)
  - `nice`, `renice`: lower the priority
  - `ulimit`: display imposed limits
  - `free`: display the memory usage

- **storage**
  - `df`: disk free
  - `du`: disk used
Managing resources

- processor resources: top, ps, kill, nice, renice
  - ps: display processes
  - top: continuously display sorted processes
  - kill: send signals to process (stop, suspend, continue)
  - nice, renice: lower the priority
  - ulimit: display imposed limits
  - free: display the memory usage

- storage
  - df: disk free
  - du: disk used
Managing resources

- processor resources: top, ps, kill, nice, renice
  - ps: display processes
  - top: continuously display sorted processes
  - kill: send signals to process (stop, suspend, continue)
  - nice, renice: lower the priority
  - ulimit: display imposed limits
  - free: display the memory usage

- storage
  - df: disk free
  - du: disk used
Managing resources

- processor resources: top, ps, kill, nice, renice
  - ps: display processes
  - top: continuously display sorted processes
  - kill: send signals to process (stop, suspend, continue)
  - nice, renice: lower the priority
  - ulimit: display imposed limits
  - free: display the memory usage

- storage
  - df: disk free
  - du: disk used
Managing resources

- processor resources: top, ps, kill, nice, renice
  - ps: display processes
  - top: continuously display sorted processes
  - kill: send signals to process (stop, suspend, continue)
  - nice, renice: lower the priority
  - ulimit: display imposed limits
  - free: display the memory usage

- storage
  - df: disk free
  - du: disk used
Managing resources

- **processor resources**: top, ps, kill, nice, renice
  - ps: display processes
  - top: continuously display sorted processes
  - kill: send signals to process (stop, suspend, continue)
  - nice, renice: lower the priority
  - ulimit: display imposed limits
  - free: display the memory usage

- **storage**
  - df: disk free
  - du: disk used
Managing resources

- processor resources: top, ps, kill, nice, renice
  - ps: display processes
  - top: continuously display sorted processes
  - kill: send signals to process (stop, suspend, continue)
  - nice, renice: lower the priority
  - ulimit: display imposed limits
  - free: display the memory usage

- storage
  - df: disk free
  - du: disk used
Managing resources

- processor resources: top, ps, kill, nice, renice
  - ps: display processes
  - top: continuously display sorted processes
  - kill: send signals to process (stop, suspend, continue)
  - nice, renice: lower the priority
  - ulimit: display imposed limits
  - free: display the memory usage

- storage
  - df: disk free
  - du: disk used
Managing resources

- processor resources: top, ps, kill, nice, renice
  - ps: display processes
  - top: continuously display sorted processes
  - kill: send signals to process (stop, suspend, continue)
  - nice, renice: lower the priority
  - ulimit: display imposed limits
  - free: display the memory usage

- storage
  - df: disk free
  - du: disk used
Organise your work before actually doing it

- meaningful naming (not lazy test, test1) but Analysis, journal/2009/08/16/log.txt
  - easy to remember after 6 months, 3 years
  - add "extensions" to files: log.txt, bday-2009.jpg not just name
  - be consistent
  - keep a daily log (cut paste command/output)
Organise your work before actually doing it

- meaningful naming (not lazy test, test1) but Analysis, journal/2009/08/16/log.txt
- easy to remember after 6 months, 3 years
  - add "extensions" to files: log.txt, bday-2009.jpg not just name
  - be consistent
  - keep a daily log (cut paste command/output)
Organise your work before actually doing it

- meaningful naming (not lazy test, test1) but Analysis, journal/2009/08/16/log.txt
- easy to remember after 6 months, 3 years
- add "extensions" to files: log.txt, bday-2009.jpg not just name
  - be consistent
- keep a daily log (cut paste command/output)
Organise your work before actually doing it

- meaningful naming (not lazy test, test1) but Analysis, journal/2009/08/16/log.txt
- easy to remember after 6 months, 3 years
- add "extensions" to files: log.txt, bday-2009.jpg not just name
- be consistent
- keep a daily log (cut paste command/output)
Organise your work before actually doing it

- meaningfull naming (not lazy test, test1) but Analysis, journal/2009/08/16/log.txt
- easy to remember after 6 months, 3 years
- add "extensions" to files: log.txt, bday-2009.jpg not just name
- be consistent
- keep a daily log (cut paste command/output)
Example

|-- DATAMANIP
|   |-- GCOMP
|       |-- MLE
|       |   |-- MLEseqnew
|       |   |       |-- MCL
|       |   |       |-- families
|       |   |       `-- FAMILIES
|       |       |-- `multmatchmle
|       |       |-- MTUseqnew
|       |       `-- MULseqnew
|   |   |-- MTU
|   |       |-- MLEseqnew
|   |       |-- MTUseqnew
|   `-- `scripts

|-- GENOMEDB
    |-- MLE
    |-- MTU
    |-- MUL
    `-- `scripts
Search for help around

- locate: find on the local disk(s)
- whatis: short info on program
- man (man man): display manual pages for a program
- find

```
[tru@callab BCGA2009]$ find . -name MUL
./GCOMP/MUL
./GENOMEDB/MUL
```
Search for help around

- **locate**: find on the local disk(s)
- **whatis**: short info on program
- **man** (man man): display manual pages for a program
- **find**

```
[tru@callab BCGA2009]$ find . -name MUL
./GCOMP/MUL
./GENOMEDB/MUL
```
Search for help around

- `locate`: find on the local disk(s)
- `whatis`: short info on program
- `man` (man man): display manual pages for a program

```bash
[tru@callab BCGA2009]$ find . -name MUL
./GCOMP/MUL
./GENOMEDB/MUL
```
Search for help around

- **locate**: find on the local disk(s)
- **whatis**: short info on program
- **man (man man)**: display manual pages for a program
- **find**

```bash
[tru@callab BCGA2009]$ find . -name MUL
./GCOMP/MUL
./GENOMEDB/MUL
```
Intermediate level knowledge of unix systems.
Avoid typing

- shell completion (tab key)
- alias, functions
  
  alias
  alias ll='ls -l'
  alias grep='grep -color=tty -d skip'
- adding directories to PATH
Avoid typing

- shell completion (tab key)
- alias, functions

```bash
alias ll='ls -l'
alias grep='grep -color=tty -d skip'
```

- adding directories to PATH
Avoid typing

- shell completion (tab key)
- alias, functions
  
  ```bash
  alias
  alias ll='ls -l'
  alias grep='grep -color=tty -d skip'
  ```
- adding directories to PATH
  
  ```bash
  PATH=$HOME/build/bin:$PATH export PATH
  ```
Avoid re-typing

- history (recall/modify the previous commands)
- shell completion (arrow keys)
- shell scripts (loops, tests, ...) - scripting instead of doing it manually
Avoid re-typing

- history (recall/modify the previous commands)
- shell completion (arrow keys)
- shell scripts (loops, tests, ...) - scripting instead of doing it manually
Avoid re-typing

- history (recall/modify the previous commands)
- shell completion (arrow keys)
- shell scripts (loops, tests, ...) - scripting instead of doing it manually
Redirecting the input, output

stdout and stdin redirections

- `> filename`
  - "create and add to" filename
  - program > filename
  - the OUTPUT file is created BEFORE anything else happens.

- `>> filename`
  - "append to" filename

- `< filename`
  - "take filename as stdin"

- `< < EOF`
  - "take all as stdin until the EOF marker"
Redirecting the input, output

stdout and stdin redirections

- `> filename`
  "create and add to" filename

- `>> filename`
  "append to" filename

  program » filename

- `< filename`
  "take filename as stdin"

- `< EOF`
  "take all as stdin until the EOF marker"
Redirecting the input, output

stdout and stdin redirections

- `> filename`
  "create and add to" filename

- `>> filename`
  "append to" filename

- `< filename`
  "take filename as stdin"

  program `< filename

- `<< EOF`
  "take all as stdin until the EOF marker"
Redirecting the input, output

stdout and stdin redirections

- `>` *filename*
  - "create and add to" *filename*

- `>>` *filename*
  - "append to" *filename*

- `<` *filename*
  - "take *filename* as stdin"

- `<<` EOF
  - "take all as stdin until the EOF marker"

```bash
program << EOF
1st command
2nd command
EOF
```
Chaining commands

- output/input with a PIPE: |
  
  program1 | program2

- one after the other with ;

- & &
  
  and

- or ||
Chaining commands

- output/input with a PIPE: |
- one after the other with ;
  
  `program1; program2`

- &&
  
  and

- or ||
Chaining commands

- output/input with a PIPE: |
- one after the other with ;
- &&
  and
  program1 && program2
- or ||
Chaining commands

- output/input with a PIPE: |
- one after the other with ;
- &&
  and
- or ||

  program1 || program2
Link and compress files, directories

- `ln, ln -s`: make hard links, soft links
- `tar`: manipulate "tar" files (tape archives: convert directory into files and backward)
- `compress/uncompress, gzip/gunzip, bzip2/bunzip2`: compression programs
- `zip/unzip, rar/unrar`: same but for the windows world
Link and compress files, directories

- `ln, ln -s`: make hard links, soft links
- `tar`: manipulate "tar" files (tape archives: convert directory into files and backward)
  - `compress/uncompress, gzip/gunzip, bzip2/bunzip2`: compression programs
  - `zip/unzip, rar/unrar`: same but for the windows world
Link and compress files, directories

- `ln, ln -s`: make hard links, soft links
- `tar`: manipulate "tar" files (tape archives: convert directory into files and backward)
- `compress/uncompress, gzip/gunzip, bzip2/bunzip2`: compression programs
- `zip/unzip, rar/unrar`: same but for the windows world
Link and compress files, directories

- `ln, ln -s`: make hard links, soft links
- `tar`: manipulate "tar" files (tape archives: convert directory into files and backward)
- `compress/uncompress, gzip/gunzip, bzip2/bunzip2`: compression programs
- `zip/unzip, rar/unrar`: same but for the windows world
Working with Linux

Tru Huynh

Avoid typing
Avoid re-typing
Redirections
Manage your disk space
Work anytime
Do not reinvent the wheel

Work anytime

- **at**: one time job at specified date/time
- **cron**: repeat jobs
- **ssh keys** instead of password
Work anytime

- at: one time job at specified date/time
- cron: repeat jobs
- ssh keys instead of password
Work anytime

- at: one time job at specified date/time
- cron: repeat jobs
- ssh keys instead of password
Use other people’s work

- **Avoid typing**
  - Avoid re-typing
  - Redirections
  - Manage your disk space
  - Work anytime

- **Do not reinvent the wheel**

- **lot of usefull commands available:**
  - grep, head, tail, cut, paste, sed, sort, uniq,
  - printf, tac, tr,

  and chaining these commands with | to avoid temporary files

- Comprehensive Perl Archive Network (CPAN):

- python’s Package Index: [http://pypi.python.org/pypi](http://pypi.python.org/pypi)
Use other people’s work

- lot of usefull commands available:
  grep, head, tail, cut, paste, sed, sort, uniq, printf, tac, tr,

  and chaining these commands with | to avoid temporary files

- Comprehensive Perl Archive Network (CPAN):
  http://www.cpan.org/

- python’s Package Index: http://pypi.python.org/pypi
Use other people’s work

- lot of usefull commands available:
  - grep, head, tail, cut, paste, sed, sort, uniq, printf, tac, tr,
  - and chaining these commands with | to avoid temporary files

- Comprehensive Perl Archive Network (CPAN):
  - http://www.cpan.org/

- python’s Package Index: http://pypi.python.org/pypi
Learn a basic NON GRAPHICAL text editor

gedit (default text editor) and nedit are nice GUI editors, but sometime you don’t/can’t have them readily available

- nano: basic text editor
- vi (esc :w :q)
- emacs (ctrl-x ctrl-c)
Learn a basic NON GRAPHICAL text editor

gedit (default text editor) and nedit are nice GUI editors, but sometime you don’t/can’t have them readily available

- nano: basic text editor
- vi (esc :w :q)
- emacs (ctrl-x ctrl-c)
Learn a basic NON GRAPHICAL text editor

gedit (default text editor) and nedit are nice GUI editors, but sometime you don’t/can’t have them readily available

- nano: basic text editor
- vi (esc :w :q)
- emacs (ctrl-x ctrl-c)
Advanced level knowledge of unix systems.
.ssh/config and tunnels

.ssh/config see also ssh_config(5):

Host ssh
HostName ssh.pasteur.fr
   LocalForward 20001 sillage.bis.pasteur.fr:22

Host sillage
HostName localhost
   Port 20001
   HostKeyAlias localhost-sillage

Host *.hku.hk
   IdentityFile ~/.ssh/id_dsa-hk2009

SOCKS tunnels:

ssh -D7070 outpost.somewhere.outhere
ssh-agent

[tru@darwin ]$ ssh callab38
Enter passphrase for key ’/home/tru/.ssh/id_dsa-hk2009’:

- ssh-agent is here to help!
- avoid typing the password key to your ssh keys!
- no reason not to have password protected keys
ssh-agent

[tru@darwin ]$ ssh callab38
Enter passphrase for key '/home/tru/.ssh/id_dsa-hk2009':

- ssh-agent is here to help!
- avoid typing the password key to your ssh keys!
- no reason not to have password protected keys
ssh-agent

[tru@darwin ]$ ssh callab38
Enter passphrase for key ’/home/tru/.ssh/id_dsa-hk2009’:

- ssh-agent is here to help!
- avoid typing the password key to your ssh keys!
- no reason not to have password protected keys
ssh-agent

[tru@darwin ]$ ssh callab38
Enter passphrase for key `/home/tru/.ssh/id_dsa-hk2009`:

- **ssh-agent is here to help!**
- **avoid typing the password key to your ssh keys!**
- **no reason not to have password protected keys**

**example:**

[tru@darwin ]$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-lyJiVx9482/agent.9482; export SSH_AUTH_SOCK;
SSH_AGENT_PID=9483; export SSH_AGENT_PID;
echo Agent pid 9483;
[tru@darwin ]$ ssh-add /home/tru/.ssh/id_dsa-hk2009
Enter passphrase for /home/tru/.ssh/id_dsa-hk2009:
[tru@darwin ]$ ssh callab38
Last login: Mon Aug 10 14:05:54 2009 from sillage.bis.pasteur.fr ...
[tru@darwin ]$ ssh-add -l
(DSA)
[tru@darwin ]$ ssh-add -d /home/tru/.ssh/id_dsa-hk2009

[tru@darwin ]$ ssh-add -l
The agent has no identities.
Managing temporary files and errors

- **fixed name**
  
  ```bash
  TMPFILE=/tmp/mytmp.file
  ```

- **PID dependant**
  
  ```bash
  TMPFILE=/tmp/mytmp.file.$$
  ```

- **PID and hostname**
  
  ```bash
  TMPFILE=/tmp/mytmp.file.hostname.$$ 
  ```

- **let the system manage it!**
  
  ```bash
  TMPFILE=mktemp
  ```
Managing temporary files and errors

- **fixed name**
  \[\text{TMPFILE}=/tmp/mytmp.file\]

- **PID dependant**
  \[\text{TMPFILE}=/tmp/mytmp.file.$$\]

- **PID and hostname**
  \[\text{TMPFILE}=/tmp/mytmp.file.hostname.$$\]

- **let the system manage it!**
  \[\text{TMPFILE}=\text{mktemp}\]

Copy your files efficiently
Managing temporary files and errors

- **fixed name**
  
  ```
  TMPFILE=/tmp/mytmp.file
  ```

- **PID dependant**
  
  ```
  TMPFILE=/tmp/mytmp.file.$$
  ```

- **PID and hostname**
  
  ```
  TMPFILE=/tmp/mytmp.file.hostname.$$ 
  ```

- **let the system manage it!**
  
  ```
  TMPFILE=mktemp 
  ```
Managing temporary files and errors

- fixed name
  ```bash
  TMPFILE=/tmp/mytmp.file
  ```

- PID dependant
  ```bash
  TMPFILE=/tmp/mytmp.file.$$  
  ```

- PID and hostname
  ```bash
  TMPFILE=/tmp/mytmp.file.hostname.$$  
  ```

- let the system manage it!
  ```bash
  TMPFILE=mktemp
  ```
rsync: keep in sync files on different machines

why not plain copy?

```
tru@callab $ scp -prv dir1 dir2 ... \\nmy.other.login@another.machine:backups
```

because rsync does it better!

- efficient: (incremental update)
- sparse file aware
- secure: uses ssh by default
- keep timestamps
rsync: keep in sync files on different machines

why not plain copy?

tru@callab $ scp -prv dir1 dir2 ... \my.other.login@another.machine:backups

because rsync does it better!

- efficient: (incremental update)
- sparse file aware
- secure: uses ssh by default
- keep timestamps
rsync: keep in sync files on different machines

why not plain copy?

tru@callab $ scp -prv dir1 dir2 ... \\ my.other.login@another.machine:backups

because rsync does it better!

- efficient: (incremental update)
- sparse file aware
- secure: uses ssh by default
- keep timestamps
rsync: keep in sync files on different machines

why not plain copy?

```
tru@callab $ scp -prv dir1 dir2 ... / my.other.login@another.machine:backups
```

because rsync does it better!

- efficient: (incremental update)
- sparse file aware
- secure: uses ssh by default
- keep timestamps
rsync: keep in sync files on different machines

why not plain copy?

```bash
tru@callab $ scp -prv dir1 dir2 ... \
my.other.login@another.machine:backups
```

because rsync does it better!

- efficient: (incremental update)
- sparse file aware
- secure: uses ssh by default
- keep timestamps

example:

```bash
tru@callab $ rsync -aPHSv dir1 dir2 ... \
-exclude '.*.swap' -exclude ... \
my.other.login@another.machine:backups
```
You already know about:

- backgrounding &
- nohup

Why another program? What else do you need?

- recover from a lost connection!
- attach/re-attach your shell from any other machine
- interact with your program EVEN after login OUT!
- cut and paste between screen's windows without mouse
- scrollback and logging

Demo!
You already know about:

- backgrounding &
- nohup

Why another program? What else do you need?

- recover from a lost connection!
- attach/re-attach your shell from any other machine
- interact with your program EVEN after login OUT!
- cut and paste between screen's windows without mouse
- scrollback and logging

Demo!
You already know about:

- backgrounding &
- nohup

Why another program? What else do you need?

- recover from a lost connection!
- attach/re-attach your shell from any other machine
- interact with your program EVEN after login OUT!
- cut and paste between screen’s windows without mouse
- scrollback and logging

Demo!
You already know about:

- backgrounding &
- nohup

Why another program? What else do you need?

- recover from a lost connection!
- attach/re-attach your shell from any other machine
- interact with your program EVEN after login OUT!
- cut and paste between screen’s windows without mouse
- scroll back and logging

Demo!
screen

You already know about:

- backgrounding &
- nohup

Why another program? What else do you need?

- recover from a lost connection!
- attach/re-attach your shell from any other machine
- interact with your program EVEN after login OUT!
- cut and paste between screen’s windows without mouse
- scrollback and logging

Demo!
You already know about:

- backgrounding &
- nohup

Why another program? What else do you need?

- recover from a lost connection!
- attach/re-attach your shell from any other machine
- interact with your program EVEN after login OUT!
- cut and paste between screen’s windows without mouse
- scrollback and logging

Demo!
screen

You already know about:

- backgrounding &
- nohup

**Why another program? What else do you need?**

- recover from a lost connection!
- attach/re-attach your shell from any other machine
- interact with your program **EVEN** after login OUT!
- cut and paste between screen’s windows without mouse
- scrollbacll and logging

- Demo!
screen

You already know about:
- backgrounding &
- nohup
- **Why another program? What else do you need?**
  - recover from a lost connection!
  - attach/re-attach your shell from any other machine
  - interact with your program EVEN after login OUT!
  - cut and paste between screen’s windows without mouse
  - scroll back and logging

- Demo!
Working with Linux

Tru Huynh

You already know about:

- backgrounding &
- nohup

Why another program? What else do you need?

- recover from a lost connection!
- attach/re-attach your shell from any other machine
- interact with your program EVEN after login OUT!
- cut and paste between screen’s windows without mouse
- scrollbar and logging

Demo!
### screen arguments:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>screen</code></td>
<td>start screen</td>
</tr>
<tr>
<td><code>screen -S myname</code></td>
<td>start &quot;mynname&quot; screen session</td>
</tr>
<tr>
<td><code>screen -ls</code></td>
<td>list screen sessions</td>
</tr>
<tr>
<td><code>screen -r screenid</code></td>
<td>re-attach screenid session</td>
</tr>
</tbody>
</table>

### inside screen (control + A)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>detach</td>
</tr>
<tr>
<td>c</td>
<td>create a new window</td>
</tr>
<tr>
<td>A</td>
<td>rename window</td>
</tr>
<tr>
<td>a</td>
<td>send a ctrl+a to the window</td>
</tr>
<tr>
<td>&quot;</td>
<td>list all window</td>
</tr>
<tr>
<td>n, &lt;space&gt;</td>
<td>go to next window</td>
</tr>
<tr>
<td>?</td>
<td>help</td>
</tr>
</tbody>
</table>
### screen cheat sheet

#### screen arguments:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>screen</code></td>
<td>start screen</td>
</tr>
<tr>
<td><code>screen -S myname</code></td>
<td>start &quot;mynname&quot; screen session</td>
</tr>
<tr>
<td><code>screen -ls</code></td>
<td>list screen sessions</td>
</tr>
<tr>
<td><code>screen -r screenid</code></td>
<td>re-attache screenid session</td>
</tr>
</tbody>
</table>

#### inside screen (control + A)

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>detach</td>
</tr>
<tr>
<td>c</td>
<td>create a new window</td>
</tr>
<tr>
<td>A</td>
<td>rename window</td>
</tr>
<tr>
<td>a</td>
<td>send a ctrl+a to the window</td>
</tr>
<tr>
<td>&quot;</td>
<td>list all window</td>
</tr>
<tr>
<td>n, &lt;space&gt;</td>
<td>go to next window</td>
</tr>
<tr>
<td>?</td>
<td>help</td>
</tr>
</tbody>
</table>
Make

- manage the dependencies of your pipeline: do not redo un-needed work

  tru@callab $ make

- example of a Makefile with all in one directory:

  ```
  # define the source/target
  SRC= $(wildcard *.inp)
  DEST = $(subst .inp,.out,$(SRC) )
  # first target is the default target
  all: $(DEST)
  # generic rule, .inp -> .out
  # $< is the name of the source file
  # $@ is the name of the target file
  %.out: %.inp
  <tab>echo processing $< to $@
  ```

- example of a Makefile for subdirectories:

  ```
  SUBDIRS = foo bar baz
  .PHONY: subdirs $(SUBDIRS)
  subdirs: $(SUBDIRS)
  $(SUBDIRS):
  <tab>$($MAKE) -C @@
  ```

- run in parallel any number of independant jobs

  tru@callab $ make -j 3
- manage the dependencies of your pipeline: do not redo un-needed work

  tru@callab $ make

- example of a Makefile with all in one directory:

```
# define the source/target
SRC= $(wildcard *.inp)
DEST = $(subst .inp,.out,$(SRC) )
# first target is the default target
all: $(DEST)
# generic rule, .inp -> .out
# $< is the name of the source file
# $@ is the name of the target file
%.out: %.inp
  echo processing $< to $@
```

- example of a Makefile for subdirectories:

```
SUBDIRS = foo bar baz
.PHONY: subdirs $(SUBDIRS)
subdirs: $(SUBDIRS)
$(SUBDIRS):
  $(MAKE) -C $@
```

- run in parallel any number of independant jobs

  tru@callab $ make -j 3
manage the dependencies of your pipeline: do not redo un-needed work

```
tru@callab $ make
```

element of a Makefile with all in one directory:

```
# define the source/target
SRC= $(wildcard *.inp)
DEST = $(subst .inp,.out,$(SRC) )
# first target is the default target
all: $(DEST)
# generic rule, .inp -> .out
# $< is the name of the source file
# $@ is the name of the target file
%.out: %.inp
  echo processing $< to $@
```

element of a Makefile for subdirectories:

```
SUBDIRS = foo bar baz
.PHONY: subdirs $(SUBDIRS)
subdirs: $(SUBDIRS)
$(SUBDIRS):
  $(MAKE) -C $@
```

run in parallel any number of independant jobs

```
tru@callab $ make -j 3
```
- manage the dependencies of your pipeline: do not redo un-needed work

  tru@callab $ make

- example of a Makefile with all in one directory:

  ```
  # define the source/target
  SRC= $(wildcard *.inp)
  DEST = $(subst .inp,.out,$(SRC) )
  # first target is the default target
  all: $(DEST)
  # generic rule, .inp -> .out
  # $< is the name of the source file
  # $@ is the name of the target file
  %.out: %.inp
  <tab>echo processing $< to $@
  ```

- example of a Makefile for subdirectories:

  ```
  SUBDIRS = foo bar baz
  .PHONY: subdirs $(SUBDIRS)
  subdirs: $(SUBDIRS)
  $(SUBDIRS):
  <tab>$ (MAKE) -C $@
  ```

- run in parallel any number of independant jobs

  tru@callab $ make -j 3
Master one editor, which ever

- nedit: ctrl + left button for rectangular selection
- vim: see "Bytes of vim"
- emacs:

Working with linux
Tru Huynh

ssh
Temporary files management
Copy your files efficiently
screen
Make
Master your text editor
Advanced shell functions
...
Master one editor, which ever

- nedit: ctrl + left button for rectangular selection
- vim: see "Bytes of vim"

- emacs:
Master one editor, which ever

- nedit: ctrl + left button for rectangular selection
- vim: see "Bytes of vim"
- emacs:
Take advantage of your shell

- **bash**: be more efficient (bash-completion)
- **zsh**: even more interactive tools, but takes time to learn
Take advantage of your shell

- bash: be more efficient (bash-completion)
- zsh: even more interactive tools, but takes time to learn
tell me :)

- `lftp` (bookmarks + scripts) and `curl` (scripts++) for http://ftp:// scriptable access
- develop projects with others
  - `rcs,cvs`
  - `subversion, mercurial, ...`
- What’s your favorite tool?
tell me :)
tell me :) 

- lftp (bookmarks + scripts) and curl (scripts++) for http://ftp:// scriptable access
- developing projects with others
  - rcs,cvs
    - subversion, mercurial, ...
- What’s your favorite tool?
tell me :)

- lftp (bookmarks + scripts) and curl (scripts++) for http://ftp:// scriptable access
- developping projects with others
  - rcs, cvs
  - subversion, mercurial, ...
- What’s your favorite tool?
tell me :)

- lftp (bookmarks + scripts) and curl (scripts++) for http://ftp:// scriptable access
- developing projects with others
  - rcs,cvs
  - subversion, mercurial, ...
- What’s your favorite tool?