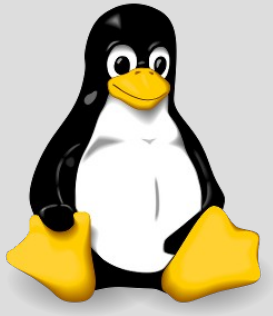
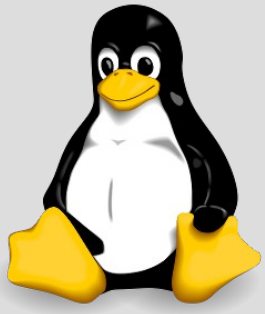
- Log into the course site
- Enter the “Lecture 2” area
- At 14:00, choose “Daily Quiz 1”
- Answer the multiple choice quiz (you have **10 minutes**)



More shell basics, get help, move around in the system



J. M. P. Alves

Laboratory of Genomics & Bioinformatics in Parasitology

Department of Parasitology, ICB, USP

Bash concepts and “tricks”

E x p a n s i o n

The Bash shell tries to expand (i.e., automatically complete) partial pieces of text or patterns that you give to it

Different kinds of expansion:

- Auto-completion (TAB key)
- Tilde expansion
- Wildcard expansion
- Brace expansion
- Arithmetic expansion
- Etc.

E x p a n s i o n

Auto-completion (TAB)

After you type a few characters ("symbols") of the command or file name you need, you can press the TAB key once and the shell will try to auto-complete as much of the name as possible. **Extremely useful!**

Some examples:

`his[TAB]` ← completes to `history`

`exp[TAB]` ← does not complete!

`exp[TAB][TAB]` ← shows possibilities (just `exp` was **ambiguous**)

this means: press TAB twice quickly

E x p a n s i o n

Arithmetic expansion

The Bash shell can be used as a simple calculator, if asked nicely:

`$((expr))` ← allows logical tests in the expression

`$[expr]` ← no tests allowed in the expression

Some examples:

```
echo $((2*30))
```
← returns 60 (2 times 30)

```
echo $[5**2]
```
← returns 25 (5 elevated to 2)

```
echo $[21%2]
```
← returns 1 (remainder of 21/2)

```
echo $((21%7))
```
← returns 0 (remainder of 21/7)

E x p a n s i o n

Wildcard file name expansion

Instead of having to type complete directory and file names many times, the shell lets you use wildcard characters and ranges to operate on multiple files at once. Also extremely useful, but has to be used with care! (specially when modifying files)

Some examples:

```
ls /etc/init.d/apache[2-]*
ls /etc/init.d/apache[2-5]*
ls /etc/init.d/apache[!2]*
ls /etc/init.d/hwc????.sh
```

returns 2 files

returns 1 file

returns 1 file

returns 1 file

Another kind of history

The **command history** is a feature of the Bash shell that is **not** an expansion, but it is even more **useful**, one could say.

The arrow keys:

- Up-arrow: previous command line used
- Down-arrow: next command line in the history
- Left-arrow: go left in the command line to edit
- Right-arrow: go right in the command line to edit

Remote access to another machine

For exercises, and so you can practice outside class, there is a virtual server that you can access from anywhere, using **`ssh`** (secure shell)

```
ssh yourname@200.144.244.172
```

(or PuTTY)

Instead of **`yourname`**, enter your **user name**

IP address: **`200.144.244.172`**

Now you do it!

Go to the course site and enter **Practical Exercise 2**

Follow the instructions to answer the questions in the exercise

Remember: in a PE, you should do things in practice before answering the question!

The wildcard characters

Like in card games, **wildcard characters** can be used in place of other cards, I mean, characters.

Main standard wildcards in the shell:

? : means any **ONE** character, exactly

***** : means any **ZERO or MORE** characters

[ ] : means a **list or range** of single characters

[!] : the **negation** of the one above (i.e., characters **NOT** in the range or list)

E x p a n s i o n

Brace expansion

This is similar to the path name expansion we saw before, so it also lets you use **wild-card characters** and **ranges**, this time in a **list of two or more**, to operate on multiple files at once. Also, now the range operator is `..` and not `-`

Some examples:

`ls /bin/d{d,f}` ← returns 2 files

`ls /bin/d{d,f,g}` ← returns 2 files and 1 error

`ls /bin/d{a..g}` ← returns 2 files and 5 errors

Expansion

Brace expansion

But wait, there's more!

Download a file with some more examples for us to play with:

```
wget http://lgbp.online/PE3.zip
```

Then run the following two commands:

```
unzip PE3.zip
cd PE3
```

E x p a n s i o n

Brace expansion

Some more examples (from inside directory PE3):

```
ls file_{1..30..3}
```
returns 10 files (from 1 to 30, every 3)

```
ls file_{1..30..2}
```
returns 15 files (the even-numbered ones)

```
ls file_{$((2*10)),$[5**2]}
```
returns 2 files (20 and 25)

But did you think I made you download that file and run all those commands just for a few simple examples? **Noooooooo...**

Now you do it!

Go to the course site and enter **Practical Exercise 3**

Follow the instructions to answer the questions in the exercise

Remember: in a PE, you should do things in practice before answering the question!

An expansion I've mentioned before but did not talk (much) about...

The tilde expansion!

- If a file path starts with a tilde (`~`), then that tilde will be expanded
- If it is `~/` (or just `~`) that means the user's own home directory
- If it is something like `~joe`, that means joe's home directory

Let's try it!

Now, run the following commands:

```
mkdir TEST somedir
scp TEST somedir
```

Something went wrong, what do I do now!?

Help in Linux systems

First and most important: **ALWAYS** carefully **read the messages** the program gave you!

Paying attention to what these messages say is often all you need in order to solve your problem.

```
scp: -r not specified; omitting
directory 'TEST'
```


Google

Usually, an Internet search is enough to solve your error...

...but it does not always tell you something **new or unexpected**!

Getting help within the system itself

(even without Internet!)

You see a command name, but don't know what the command does

`whatis`

Displays one-line manual page descriptions

The **whatis** command, followed by your query, tells us a short description of the command's purpose.

list directory contents

Try it!

PE4!

`ls`	`echo`
`cp`	`history`
`wget`	`export`
`scp`	`cd`

What if I do not even know what command to use?

apropos

Searches the manual page names and descriptions for keywords given by the user

Running `apropos word` searches the documentation system for **`word`** and presents the results (for exact searches, use the **`-e`** option, e.g., `apropos -e word`)

Try it! Which command:

```
mv (1) - move (rename) files
```

- **moves or renames files?**
- **merges lines of files?**
- **does secure file copy?**
- **copies files and directories (local)?**
- **prints lines matching a pattern?**

PE5!

OK, so you found the command that does what you want. Now what?

man & info

There are files in the system that contain the **manual** or the **information** (man and info pages, for short) for the program; not all programs have this – it depends on the developer (program creator)

- Man pages are usually drier, more technical (but not always! again, depends on the author)
- Info pages usually contain more text, explain things in a longer and friendlier way, and resemble a browser, with several pages to navigate

Let's try it!

```
man ls
info ls
```

What if I just want a quicker reminder of the options and switches the program can take?

...?

Actually, there is **no program** in this case; each program has its own mechanism, or even none at all! (depends on the author) Good programs have **built-in help**, accessible using the `--help` (or sometimes `-h` or `-help`) switches after the command

- The built-in help is usually the **most succinct** of the help mechanisms in the system (again, depends on the author)

- If you are already used to the way the command works, it might be all you need though!

Let's try it!

```
ls --help
bash -help
cd --help
```

- Depending on the distro, shell built-in command help is accessed using (for example): `help cd`

On a CentOS 7 system...

```
[jmalves@igdrasil ~]$ cd --help
bash: cd: --: invalid option
cd: usage: cd [-L|[-P [-e]]] [dir]

[jmalves@igdrasil ~]$ help cd
cd: cd [-L|[-P [-e]]] [dir]
    Change the shell working directory.

    Change the current directory to DIR.
… … …
```

cd is a built-in command!

Now you do it!

Go to the course site and enter **Practical Exercise 6**

Follow the instructions to answer the questions in the exercise

Remember: in a PE, you should do things in practice before answering the question!

But always remember: *a good web search, performed in a* ***smart way****, can save you a lot of time!* ***Many people have already asked*** *about the problem you have, and* ***many people have given different answers*** *(sometimes with very detailed explanations); it is almost certain that one of them will suit your situation.*

Recap

- There are many string expansion possibilities to make your life a bit easier when typing in the shell
- The **`TAB`** key and **tilde** expansions can save you a ton of time
- There is also a command history accessible using the **arrow keys**
- Help can be found **online** (Google et al.), or **within the system** itself:
    - Internal help (**`--help`**, **`-h`** etc. or **`help`** for shell built-in commands)
    - **`whatis`**
    - **`apropos`**
    - **`man`**
    - **`info`**