SCITAS: Cheat Sheet

Unix commands

Help and info

`man`	get help on commands
`apropos`	search for a command and print a brief description
`echo`	print the arguments in the terminal

File manipulation

`ls`	list files
`cd`	change directory
`pwd`	path of the current
`cp`	copy files
`mv`	move/rename files
`rm`	remove file caution no trash
`mkdir`	create a folder
`find`	find a file

File edition

`cat`	concatenate files, used to print the content of file
`touch`	update the date of a file, used to create empty files
`tail/head`	print the end/beginning of a file
`more/less`	print the content of a file "more or less the same"
`grep`	search a string in a file
`vi`	`i` switch from command to insert mode `esc` go back from insert to command mode `:wq` save and quit `:q!` force quit
`emacs`	`Ctrl-x Ctrl-s` save file `Ctrl-x Ctrl-c` quit
`nano`	`Ctrl-o` save file `Ctrl-x` quit

File permissions

`rw-` `r--` `r--` `richart scitas-ge CodingStyle`

- `u` User permissions
- `g` Group permissions
- `o` Other users permissions

`-` or 0	Permission not granted
`r` or 4	Read/List permission
`w` or 2	Write/Create permission
`x` or 1	Execute/Traverse permission

`chmod`	change permissions
`chgrp`	change main group
`chown`	change ownership
`newgrp`	login with a new group as main group
`id`	print user and group ids

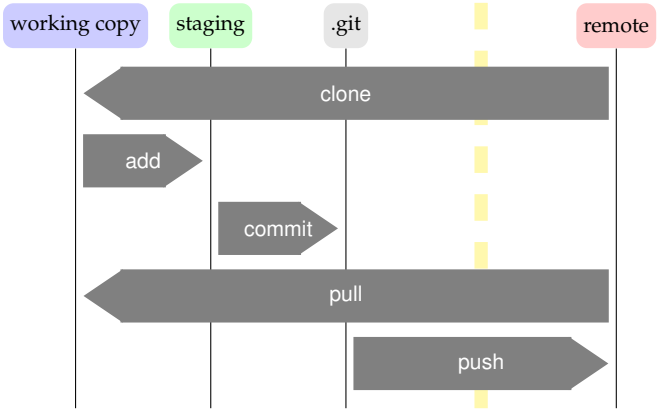
Process manipulation

`ps`	list the running processes
`jobs`	list the jobs running in the current shell
`kill`	kill processes
`command &`	runs the command in background
`Ctrl-z`	stops the current foreground command
`bg`	sends job in background
`fg`	pull job in foreground

Redirections

`command > file`	redirect the standard output to `file`
`command 2> file`	redirect the standard error to `file`
`command < file`	redirect `file` to the standard input
`command1 | command2`	redirect the stdout of `command1` in stdin of `command2`

Versioning : GIT



Basic usage

Cloning a repository	`git clone <url>`
Adding/staging a file	`git add <file>...`
Commiting staged changes	`git commit -m "comment"`
Pushing to a remote	`git push`
Pulling from a remote	`git pull`
Seeing local modifications	`git status`

Branching

Creating a branch	`git checkout -b <branch>`
Changing current branch	`git checkout <branch>`
Name of current branch	`git branch`
List all branches	`git branch -vv -a`
Merging a branch in the current one	`git merge <branch>`
Pushing a branch to a remote	`git push <remote> <branch>`
Pushing a branch to a remote, and tracking the remote branch	`git push -u <remote> <branch>`
Pulling/Merging a remote branch in the current branch	`git pull <remote> <branch>`

Remotes

Adding a remote	`git remote add <name> <url>`
List remotes	`git remote -v`

Back in history

Canceling uncommited changes	`git checkout -- <file>...`
Canceling last unpushed commit, and keeping modifications	`git reset HEAD~`
Canceling last unpushed commit, and loosing the modifications	`git reset --hard HEAD~`
Generate a canceling commit for a given pushed commit	`git revert <commit>`

Modules

`module avail`	list all possible modules
`module load <module>`	load a module
`module unload <module>`	unload a module
`module purge`	unload all the modules
`module list`	list all loaded modules

Job Scheduling : Slurm

General commands

Submitting a job	`sbatch`
Allocating resources for interactive jobs	`salloc`
Visualize the queue state	`squeue`
Visualize the shares	`sshare`
List the partitions	`sinfo`

Options common to all commands

`-A` `--account`	defines which account to use For this class you are in `scitas-courses`
`-u` `--user`	defines the user mainly useful for `squeue`
`--reservation`	defines which reservation to use For this class a reservation named `mpi_course` is set on `deneb`
`-p` `--partition`	defines the partition to use

Options for `sbatch` / `salloc`

`-N` `--nodes`	defines the number of nodes to use
`-t` `--time`	defines the maximum wall time
`-n` `--ntasks`	number of tasks, in the MPI sense
`-c` `--cpus-per-task`	number of cpus per process
`--ntasks-per-node`	number of tasks per node
`--mem`	defines the quantity of memory per node requested

EPFL specific commands

Visualize your jobs	`Squeue`
Get info on a job	`Sjob <jobid>`
Visualize the shares of your groups	`Sshare`

Example of job script

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --ntasks 1
#SBATCH --cpus-per-task 1
#SBATCH --time 0:10:0
#SBATCH --reservation mpi_course
#SBATCH --account scitas-courses

module purge
module load intel

export OMP_NUM_THREADS=${
    SLURM_CPUS_PER_TASK}

srun ./my_code
```

OpenMP

Include `omp.h`
Compiler options

`intel` ver < 15	compile and link with `-openmp`
`intel` ver $>= 15$	compile and link with `-qopenmp`
`gcc`	compile and link with `-fopenmp`

MPI

Include `mpi.h`

On EPFL clusters two implementation of MPI can be found, `mvapich2` and `intelmpi`

Compiler wrappers:

`intelmpi`	`mpiicc`, `mpiifort`, `mpiicpc`
`mvapich2`	`mpicc`, `mpif90`, `mpicxx`

The wrapper to run mpi codes is `srun`, this automatically uses the parameter passed to SLURM for the number of tasks.