Linux & ROS Cheat Sheet

Author: Yize Wang
Institute: Autonomous Systems Lab
Last Updated: April 14, 2021

File Commands

$ ls	list contents of the current directory
$ ls -al	list hidden contents of the current directory
$ cd	change the directory to home
$ cd -	change the directory to the previous one
$ cd $DIR	change the directory to $DIR
$ mkdir $DIR	make a new directory named $DIR
$ pwd	print the working directory
$ rm $FILE	remove $FILE
$ rm -r $DIR	remove $DIR recursively
$ rm -f $FILE	force remove $FILE
$ rm -rf $DIR	force remove $DIR recursively
$ cp $FILE1 $FILE2/$DIR	copy $FILE1 to $FILE2/$DIR
$ cp -r $DIR1 $DIR2	copy $DIR1 to $DIR2 recursively
$ mv $FILE1 $FILE2/$DIR	move $FILE1 to $FILE2/$DIR
$ ln -s $FILE $LINK	create a symbolic link $LINK to $FILE
$ touch $FILE	create $FILE
$ cat $FILE	view content of $FILE
$ cat > $FILE	write input into $FILE
$ echo $STRING/$VAR	print $STRING/value of $VAR
$ more $FILE	print content of $FILE
$ head $FILE	print the first 10 lines of $FILE
$ tail $FILE	print the last 10 lines of $FILE
$ gedit $FILE	edit $FILE using GUI text editor
$ vim $FILE	edit $FILE using Vim

System Information

$ env		print environment variables
$ date		print system date and time
$ man $COMMAND		print user manual of $COMMAND
$ whereis $APP		print locations of $APP
$ which $APP		print executable file of $APP
$ ps		print process status
$ ps -aux		print all running process
$ htop		print currently running processes and more
path symbolic links	.	current directory
	..	parent directory
	~	home directory
	/	root directory
output direction	>	to a file (rewrite)
	>>	to a file (append)
	|	pipe output of first command to second

Linux Shell

Ctrl+C	kill the current process
Ctrl+Z	suspend the current process
$ fg	resume the suspended process in foreground
$ bg	resume the suspended process in background
Ctrl+W	erase one word in the current line
Ctrl+U	erase the whole current line
Ctrl+R	reverse search in the previous commands
Ctrl+A	go to the beginning of the line
Ctrl+E	go to the end of the line
Ctrl+D	log out of the current session
$ exit	log out of the current session
$ clear	clear the terminal screen

Use **Ctrl+R** to reverse search, type part of a command and hit **Ctrl+R** repeatedly. **Ctrl+A** is especially useful when you forget to add sudo before the command.

Git

$ git clone $URL	clone the repository from $URL
$ git status	print current branch status $BRANCH
$ git branch $BRANCH	create a new branch named $BRANCH
$ git checkout $BRANCH	switch to the branch named $BRANCH
$ git merge $BRANCH	combine $BRANCH into the current one
$ git fetch	download all history from GitHub
$ git merge	combine remote branches into local branch
$ git push	upload all local branch commits to GitHub
$ git pull	update local branch from GitHub
$ git log	list version history for current branch
$ git log --follow $FILE	list version history for $FILE
$ git show $COMMIT	output content changes of $COMMIT
$ git add $FILE	stage $FILE
$ git commit -m "$MESSAGE"	commit staged file with $MESSAGE
$ git reset $FILE	reset $FILE
$ git reset --hard	reset all uncommitted changes
$ git clean -fd	recursively force remove unstaged files

Secure Shell (SSH)

$ ssh $USER@$HOST	connect $HOST as $USER
$ ssh $IP_ADDRESS	connect $IP_ADDRESS
$ ssh -p $PORT $USER@$HOST	connect $HOST on $PORT as $USER
$ ssh-copy-id $USER@$HOST	add the key to $HOST as $USER

Package

$ sudo apt-get update	synchronize package index files from sources
$ sudo apt-get upgrade	install latest versions of installed packages
$ sudo apt-get install $PACKAGE	install $PACKAGE
$ sudo dpkg -i $PACKAGE.deb	install a Debian package $PACKAGE.deb
$ ./configure	configure building settings
$ make	build the program from source code
$ make install	install the program

Terminator

Ctrl+Shift+I	open a new window
Ctrl+Shift+T	open a new tab
Ctrl+Shift+E	split terminals vertically
Ctrl+Shift+O	split terminals horizontally
Alt+<arrow key>	switch to a different terminal

Terminal Multiplexer (TMUX)

$ tmux	start TMUX
$ tmux ls	list all sessions
$ tmux a -t $SESSION_NAME	attach to $SESSION_NAME
$ tmux new -s [$SESSION_NAME]	create a new session with $SESSION_NAME
Ctrl+B	prefix
Prefix+%	split terminals horizontally
Prefix+"	split terminals vertically
Prefix+<arrow key>	switch to a different terminal
Prefix+C	create a new window in current session
Prefix+$NUM	switch to $NUM window
Prefix+D	detach from the current session

Searching

$ grep $PATTERN $FILES	search for $PATTERN in $FILES
$ grep -r $PATTERN $DIR	search for $PATTERN recursively in $DIR
$ grep -n $PATTERN $FILES	search for $PATTERN and print line numbers
$ grep -C1 $PATTERN $FILES	search for $PATTERN and print 1-line context
$ $CMD | grep $PATTERN	search for $PATTERN in $CMD's output
$ sudo updatedb	update searching database for **locate** command
$ locate -b $PATTERN	find files and dirs containing $PATTERN

Docker

$ docker build -t $IMAGE:$TAG	build $IMAGE with tag $TAG
$ docker image ls	list all local images with Docker Engine
$ docker image rm $IMAGE:$TAG	delete image from local image store

$ docker tag $OLD_IMAGE:$OLD_TAG $REGISTRY/$NEW_IMAGE:$NEW_TAG
retag local image with new image name and tag
Eg. $ docker tag myimage:1.0 myrepo/myimage:2.0

$ docker push $REGISTRT/$IMAGE:$TAG
push image to registry
Eg. $ docker push myrepo/myimage:2.0

Eg. $ docker container run –name web -p 5000:80 alpine:3.9
run container from Alpine version 3.9 image, name the running container "web" and expose port 5000 externally, mapped to port 80 inside the container

Eg. $ docker container stop/kill web
stop "web" container through SIGTERM/SIGKILL

Miscellaneous

$ sudo $COMMAND	run $COMMAND with elevated privilege
$ $COMMAND --help	print $COMMAND's usage help
$ ip address	print all internet protocol addresses
$ ping $HOST	ping $HOST and print results
$ tar xfz $FILE.tar.gz	extract files from $FILE.tar.gz

ROS Catkin Workspace

$ **roscd $PACKAGE**	change directory to $PACKAGE's location
$ **catkin build**	build the whole workspace
$ **catkin build $PACKAGE**	build $PACKAGE
$ **catkin clean**	clean the whole workspace
$ **catkin config $OPTIONS**	configure catkin workspace with $OPTIONS
$ **wstool init**	set up current directory as workspace
$ **wstool merge $ROSINSTALL**	merge $ROSINSTALL into the workspace
$ **wstool up**	update configuration elements

Always remember to **$ source ~/catkin_ws/devel/setup.bash**.

ROS Run

$ **roscore**	invoke the core of ROS
$ **roslaunch $PACKAGE $LAUNCHFILE**	launch $LAUNCHFILE in $PACKAGE

$ **rosrun $PACKAGE $EXECUTABLE** [**$PARAM**:=**$VALUE**]
run node $EXECUTABLE from $PACKAGE [with $PARAM set to $VALUE]
Eg. $ rosrun rviz rviz -d maplab.rviz

ROS Node

$ **rosnode ping $NODE**	test connectivity to $NODE
$ **rosnode list**	list active nodes
$ **rosnode info $NODE**	print information about $NODE
$ **rosnode machine**	list nodes running on the machine
$ **rosnode kill $NODE**	kill the running $NODE

ROS Parameter

$ **rosparam list**	list all parameter names
$ **rosparam set $PARAM $VAL**	set value of $PARAM to $VAL
$ **rosparam get $PARAM**	print value of $PARAM
$ **rosparam load $YAML**	load parameters from $YAML
$ **rosparam dump $YAML**	dump parameters to $YAML
$ **rosparam delete $PARAM**	delete $PARAM

ROS Topic

$ **rostopic list**	print information about active topics
$ **rostopic bw $TOPIC**	display bandwidth used by $TOPIC
$ **rostopic echo $TOPIC**	print messages from $TOPIC
$ **rostopic find $TYPE**	find topics of $TYPE
$ **rostopic hz $TOPIC**	display publishing rate of $TOPIC
$ **rostopic info $TOPIC**	print information about $TOPIC
$ **rostopic pub $TOPIC**	publish data to $TOPIC
$ **rostopic type $TOPIC**	print type of $TOPIC
$ **rosmsg show $TYPE**	print structure of $TYPE

ROS Service

$ **rosservice list**	list active services
$ **rosservice call $SERVICE $ARGS**	call $SERVICE with $ARGS
$ **rosservice find $TYPE**	find services of $TYPE
$ **rosservice info $SERVICE**	print information about $SERVICE
$ **rosservice type $SERVICE**	print type of $SERVICE
$ **rosservice uri $SERVICE**	print uri of $SERVICE
$ **rossrv show $TYPE**	print structure of $TYPE

ROS Environmental Variables

ROS_ROOT	location of core ROS packages
ROS_MASTER_URI	location of the master
ROS_PACKAGE_PATH	location for more ROS packages
ROS_HOSTNAME	network address of a node
ROS_IP	IP address of a node

ROS Bag

$ **rosbag record $TOPIC**	record $TOPIC into bag
$ **rosbag info $BAG**	print content summary of $BAG
$ **rosbag play $BAG**	play back content of $BAG
$ **rosbag check $BAG**	check play-ability of $BAG in current system
$ **rosbag compress $BAG**	compress $BAG using BZ2
$ **rosbag decompress $BAG**	decompress $BAG using BZ2

When simulating in ROS, remember **$ set use_sim_time true** and to append **--clock**.

ROS Visualization Tools

$ **rviz**	3D visualization of data and models
$ **gzclient**	Gazebo GUI
$ **rqt**	powerful GUI tool
$ **rqt_plot**	simple and lightweight plotting
$ **rqt_bag**	visualize content of a bag
$ **rqt_image_view**	visualize camera images
$ **rqt_graph**	visualize computation graph
$ **rqt_tf_tree**	visualize TF frame tree

ROS Packge Structure

package.xml	manifest, dependencies and plugins
CMakeLists.txt	description of compilation procedure
src/	C and C++ source codes
build/	generated makefiles and support files
devel/	compiled binaries, libraries, headers
include/	C and C++ header files
scripts/	Python and bash scripts
config/	YMAL configuration files
cfg/	dynamic reconfigure scripts
launch/	launch files

ROS TF2

$ **rosrun tf tf_echo $FRAME1 $FRAME2**
print coordinate frame relationship between $FRAME1 and $FRAME2
Eg. $ rosrun tf tf_echo /map /odom

$ **rosrun tf view frames**	visualize coordinate transform tree

tf2 is a power package to deal with coordinate **t**rans**f**orm. It maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

ROS Launch File

<node name=$NODE pkg=$PACKAGE type=$EXE [args=$ARGS]/>
launch $NODE using the $EXE from $PACKAGE with command line arguments $ARGS
Eg. <node name="rosbag_record" pkg="rosbag" type="record" args="-a" output="screen"/>

<include file=$LAUNCHFILE/>	import $LAUNCHFILE into the current one

Eg. <include file="$(smb_local_planner)/launch/local_planner.launch"/>

<arg name=$ARG/>	declare the existence of $ARG
<arg name=$ARG value=$VAL/>	declare $ARG with constant value $VAL
<arg name=$ARG default=$VAL/>	declare $ARG with default value $VAL

Eg. <arg name="rviz" value="true"/>

<param name=$PARAM value=$VAL/>	set $PARAM to $VAL

Eg. <param name="frequency" value="300"/>

<remap from=$OLD to=$NEW/>	remap name $OLD to name $NEW

Eg. <remap from="/base_pose_measured" to="/base_pose_measured_disabled"/>