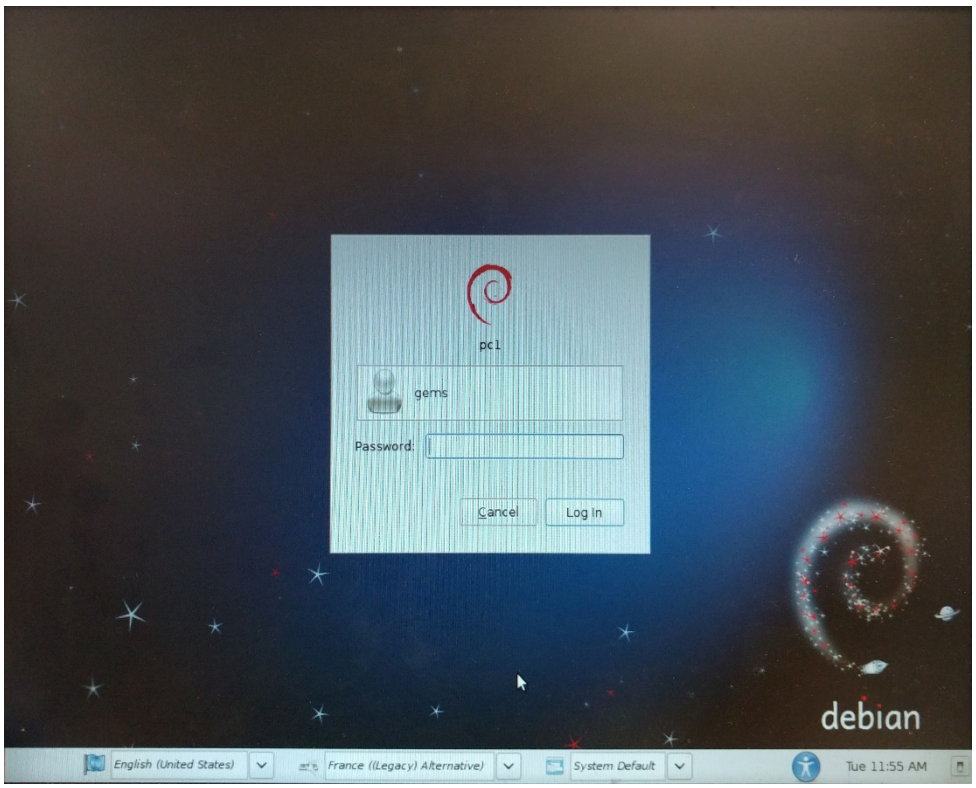
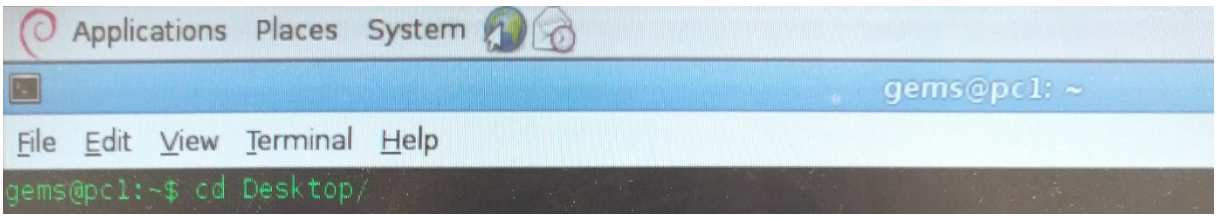
How to use Linux machines

I. Getting started

When you first boot up you will have to log-in on all the machines. For each the **username** is **"gems"** and the **password** is **"gems"**.



Before any manipulation you will have to **install the driver for the CAN interface ON EACH MACHINE**. To do so, open a command prompt "Terminal", and navigate to the "Desktop" folder using the command "*cd Desktop*".



Then, go to the folder named "Janus-CAN-Linux-driver-BETA/driver" (again using "*cd*"). To install the driver, type "*sudo ./install.sh*". You should be asked to enter the admin password, which is "gems".

Once you have done that for each machine you can start to work.

II. Useful command operations and descriptions

When you are typing the name of a file or a folder in the terminal, you can press "**Tab**" in order to **auto-complete** the name. If several files begin with the same characters, then you can quickly press "Tab" to times, and the terminal will display all the files beginning with these characters.
This **feature** is **useful** when you will use the cd, ./, cp, mv, etc commands described below.

1. Command operations

List files that contains a set of characters :
You can use "*ls | grep characters_to_find*" in order to list all the files whose the name contain the characters you want to find.

List all processes running on your session :
You can use "*ps -ef | grep name_of_the_process*" in order to list the processes running on your session "gems" whose the name contain the characters you have specified.

Killing a process manually :
After the use of "*ps*" above, you van kill the process found by using "*kill -9 id_of_the_process*".

Plugging a USB drive :
In order to mount a USB drive on a computer, you have to **disconnect the mouse**, and **replace it with the USB drive**. When you are done, **disconnect safely the USB drive and reconnect the mouse.**

2. Command descriptions

cd : This command stands for "**Change Directory**", and take for argument the path of the directory you want, relatively to the current folder. You can also use "*cd*" by specifying the path from the root. Please note that "*cd /Desktop*" and "*cd Desktop*" are not the same commands. Only the second one will work, if you are in /home/gems.
Special examples :
- "*cd ~/[path/to/your/directory]*" corresponds to "*cd /home/gems/[path/to/your/directory]*".
- "*cd ..*" will send you to the parent directory.
- "*cd -*" will send you to the previous path you were.

sudo : This command allows to **execute a command with the administrator privileges**. If you use this, you should always be asked for your **admin password**, which is "**gems**". You will usually use this command with the command *./*, to execute an executable or a shell script ([files].sh).
If you do "***sudo su*"**, you will be logged as an **administrator during the whole terminal session**, until you run the command "*exit*", or you close the terminal.

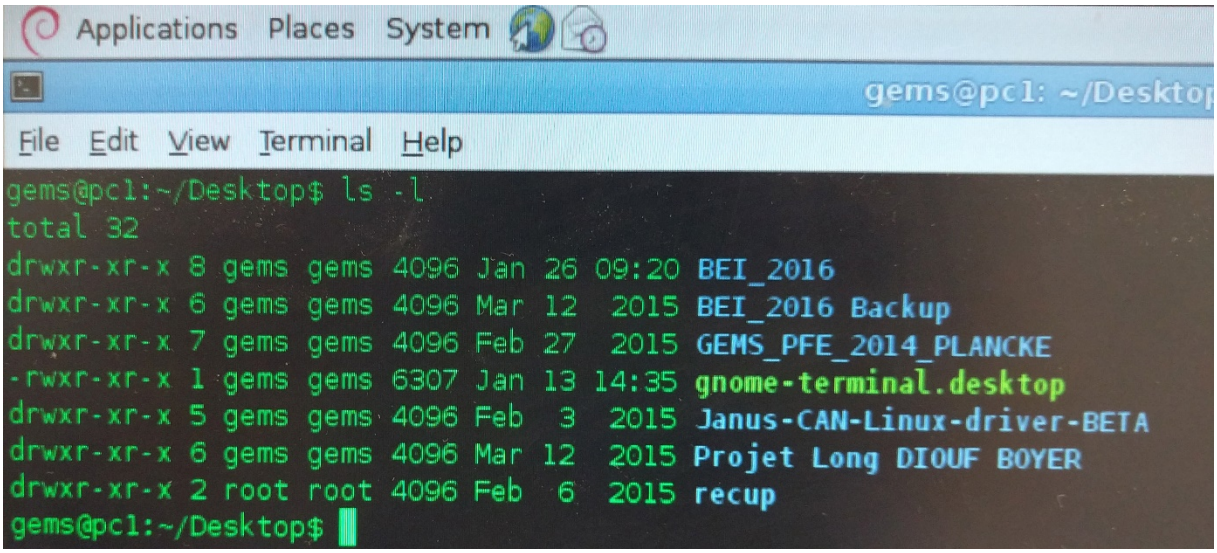
./ : This command permits to **execute** an executable (for example, a compilation you have done with the command "*make*") or a shell script ([files].sh).

cp and **mv** : These commands stands for "**Copy**" and "**Move**". Both take 2 arguments, which are the path of the original file, and the path of the destination. If you want to make a copy of one folder and its content, you have to add the **option "-r"** in order to do it **recursively**.

Special examples :
- “cp -r /home/gems/Desktop/BEI_2017-2018/ /home/gems/Desktop/Copy_of_BEI_2017-2018/” will make a copy of the folder “BEI_2017-2018” and its content to “Copy_of_BEI_2017-2018”.
- “mv /home/gems/Desktop/BEI_2017-2018/ /home/gems/Desktop/Renamed_BEI_2017-2018/” will rename the folder “BEI_2017-2018” to a new folder “Renamed_Copy_of_BEI_2017-2018”.

mkdir and **rmdir :** These commands stands for “**Make Directory**” and “**Remove Directory**”, so they respectively create and delete a directory. Please note that the directory has to be empty in order to be removed.

ls : This command stands for “**List directory contents”**. It lists the content of the directory you are. The command “**ls -l**” lists the same content, but in long listing fashion (with permissions, etc).



rm : This command stands for “**Remove”,** so it will remove all the files in argument. You will be asked to confirm your choice, but you can **force the deletion** with the **option “-f”**. If you want to delete a folder and its content, you also have to add the **option “-r”** to do it **recursively**.
Special examples :
- “*rm -rf ~/Dekstop/BEI_2017-2018/**” will remove all the content of the folder “BEI_2017-2018”. Please note that the star “*” means “anything”, so BE CAREFUL.
- “*rm -f /home/gems/Dekstop/BEI_2017-2018/*.c*” will remove all the files with the “.c” extension in the folder “BEI_2017-2018”.
- “*rm -rf /home/gems/Dekstop/BEI_2017-2018/*” will remove the folder “BEI_2017-2018” and its content.

fdisk -l : List all the **drives connected to the computer**.

mount and **umount :** These commands are used to **mount and unmount** a USB **drive**, a Hard-Disk Drive, a Solid-State Drive, etc… Both take 2 arguments, which are the path of the drive, and the path of the directory where the files will be accessible.

reboot and **poweroff** : No need to describe these ones ;)
man : Open the **manual page** of the command given in argument. Press “q” to quit.

III. Compilation process

Reminder :
After being written, a C (or C++) code has to be compiled in order to make an executable file. This compilation process can be done using software like CodeBlock or Visual C/C++ (Windows) or Xcode (Mac OS). All these software will just use gcc (or g++) (GNU Code Compiler) to compile your project, but you can do it by your own thanks to a makefile, whose structure is presented below.

```
all : ert_main.o simulink_testcode.o saveData.o clearData.o
	g++ ert_main.o simulink_testcode.o saveData.o clearData.o -o simulation

ert_main.o :
	g++ -c ert_main.cpp -o ert_main.o

simulink_testcode.o :
	g++ -c simulink_testcode.cpp -o simulink_testcode.o

saveData.o :
	g++ -c saveData.cpp -o saveData.o

clearData.o :
	g++ -c clearData.cpp -o clearData.o

clean :
	rm *.o

mrproper : clean
	rm simulation
```

make *.o : compile the correspondent *.cpp.

make or **make all** : compile all the code and create an executable file. It first calls each file compilation procedure, and then link all the objects together in order to make an executable file.

make clean : clean the directory by erasing all the ".o". This could be use if the executable does the same things whereas you have compiled a new code.

make mrproper : the same as "make clean", but erase also the executable file.

When you are in your project directory (with the code and the makefile), you can call the appropriate make command to do what you need.