Physics Linux System User's Guide

A short introduction to the most commonly used Unix (Linux) commands. For more info, read a book[1] or read the info pages.

[1]A good starting point is *The Unix Programming Environment* by Brian W. Kernighan and Rob Pike. Prentice Hall, Inc., 1984 or *Linux in a Nutshell, 4th Edition A Desktop Quick Reference* By Ellen Siever, Stephen Figgins, Aaron Weber, 4th Edition June 2003

Welcome to McGill.
Before we start, let's talk about a few basic rules you must respect.

- The computer on your desktop is *not yours*. It belongs to the Department. It is loaned to you so that you can do physics. Please do not abuse this privilege.

- If any serious problem occurs, do not try to deal with it yourself. *Contact the operators at* `operators@physics.mcgill.ca` (Paul and Juan).

- **Never** turn off, reset or reboot your computer. Our computers are shared resources. Others may be running jobs on your desktop. In case of emergency, contact the system operators.

- The above principles also apply to the printers. In addition, our printers need special transparency films. **Never** use ordinary transparency films in the printers. It *will* jam. Ask the system operators for the transparency films.

- **Never** unplug anything or plug in anything without permission.

- If you bring your laptop, you *must* ask your supervisor to get you a new ethernet socket to connect to the Net. **Never** unplug the ethernet connection from your desktop. In addition, if your laptop is running Windows, it is your responsibility to make sure that it is properly firewalled and updated constantly with the latest Microsoft patches.

- You are not allowed to set up switches, hubs or your own wireless networks.

- Do not abuse your disk space. Keep only the necessary files on your disk. Personal items should be promptly transferred to your own computer.

Quick Reference

Changing Password

- `yppasswd` : Changes your password.

Information

- `info` : To find out about a command.

  (ex) `info lpr`

  To search info pages for a keyword

  `info --apropos=keyword`

- `man` : To read a manual page about a command.

  (ex) `man lpr`

Directory related commands

- `ls` : Lists files and subdirectories in the present directory.
- `mkdir` : To make a directory.

  (ex) `mkdir sub_dir`

- `rmdir` : To remove an *empty* directory.

  (ex) `rmdir sub_dir`

- `cd` : "change directory"

  (ex) `cd sub_dir`

- `pwd` : "print working directory"

File & Directory related commands

- `cp` : To copy a file or a directory.

  (ex) `cp original copy`

  (ex) `cp -r original_dir copy_dir` (Be careful with this!)

- `mv` : To move a file or a directory.

  (ex) `mv old_name new_name`

- `rm` : To remove a file.

  (ex) `rm file`

- `more` or `less` : To read a file.

  (ex) `less unix.txt`

- `nedit`, `kdedit`, `vi`, `emacs` : To edit a file.

  (ex) `nedit unix.txt`

- `tar` : To collect a number of files and directories in one big file.

  To make : `tar cvf all_files.tar file1 file2 files*`

  To un-tar : `tar xvf all_files.tar`

- `gzip` or `bzip2` : To compress a big file.

  (ex) `gzip all.tar`

- `gunzip` or `bunzip2` : To decompress a zipped file.

  (ex) `gunzip all.tar.gz`

Compiling and running programs

- `gcc` or `cc` : C Compiler.

  (ex) `gcc -o prog_name file.c files.*.c -lm`

- `g77` or `f77` : Fortran 77 Compiler.

  (ex) `g77 -o prog_name file.f files.*.f`

- `qsub` : To submit to the batch que.

- `<ctrl>c` : To interrupt a running program.

- `<ctrl>z` : To suspend a running program.

- `bg` : To run the suspended job in the background.

- `fg` : To run the suspended job in the foreground or to bring a job in the background to the foreground.
- `jobs` : To list your jobs running in the background.
- `ps -u` : To list all of your processes
- `kill` : To kill a job or a process

LaTeX, PostScript, PDF, Printing

- `latex` : To latex a LaTeXfile.

  (ex) `latex unix.tex`
- `xdvi` : To preview a `.dvi` file on an X-terminal.

  (ex) `xdvi unix.dvi`
- `dvips` : To make a PostScript (ps) file from a dvi file to preview or to print.

  (ex) `dvips unix.dvi -o unix.ps`
- `gv`: To preview a ps file or a pdf file.

  (ex) `gv unix.ps`
- `dvips -Ppdf` : To make a pdf-ready ps file

  (ex) `dvips -Ppdf unix.dvi -o unix.ps`
- `ps2pdf`: To make a pdf file from a ps file.

  (ex) `ps2pdf unix.ps`
- `acroread, xpdf, gv`: To preview and print a pdf file.

  (ex) `acroread unix.pdf`

  (ex) `xpdf unix.pdf`
- `lpr` : To print a PostScript file or a text file.

  (ex) `lpr unix.ps`

- `lpstat` or `lpq` : To ask a printer how it is doing.

E-mail, WWW, Connecting

- `pine` : To read and send e-mails.
  - To read an email, just `enter`.
  - To 'compose' a new email, press `c`.
  - To 'reply', press `r`.
  - To 'delete', press `d`.
  - To 'save', press `s`.
  - To change levels, use `<, >`
  - To send files, use 'attachment'.
  - To quit `pine`, press `q`.
- `mail person@some.where.com < file` : To send an *ascii* file without going through an e-mail program.
- `mozilla &` : To run the web-browser Mozilla in the background
- `~/WWW` : Your McGill homepage location if you are with HEP/TNP/ASTRO.
- `/WWW/user_name` : Your McGill homepage location if you are with CMP/GANG.
- `ssh user_name@other.machine.some.where` : To securely login to other computers. Also use this to connect from home.
- `scp` : To securely copy a file from one machine to another.

  (ex) `scp file_1 user_name@other.machine.some.where:file_2`

  (ex) `scp user_name@other.machine.some.where:file_1 file_2`
- `sftp` : To securely transfer files from one account another

- To get your mails from outside McGill: Log in using `ssh` and do `pine`.

  Or, configure your browser/outlook/thunderbird with IMAP with SSL and set the Incoming server to `imap.physics.mcgill.ca` port number 993. Use secure connection type SSL. Your out going sever (SMTP) should be your ISP's server. In the Department, this is `mx1.hep.physics.mcgill.ca` port number 587, secure connection type TLS.

Other useful commands and programs

- `ypchsh` : To switch your shell globally. Default is `tcsh`.
- `chmod` : To change the attribute of a file or a directory.

  Usage: `chmod [ugoa][+-][rwx] file`
- `df` : To check filesystem disk space usage
- `du` : To check your own disk space usage
- `convert` : To convert one graphics format to another.

  (ex) `convert picture.eps picture.jpg`
- `display` : To view a graphics file

  (ex) `display picture.jpg`
- `nice` : To be nice to other people using the same machine

  (ex) `nice convert huge_picture.eps huge_picture.jpg`
- `<` : Treat the lines in a file as keyboard inputs (STDIN: standard input).

  (ex) `prog < input_dat`
- `>` : To redirect the standard output (STDOUT) of a program to a file.

  (ex) `prog < input_data > output_data`
- `>>` : To append the output of a program to a file.

  (ex) `prog < input_data >> output_data`
- `|` : Pipe. To pass STDOUT as a file.

  (ex) `grep verb unix.tex | less`
- `grep` : To globally search for a regular expression and print the lines containing it.

  (ex) `grep verb unix.tex`

1 Getting Started

McGill physics machines mostly run Debian Linux with KDE window managers. So sitting in front of an open terminal, the first thing you see is the LOGIN window. Interface is intuitive. Just type in your user-name and password and you are logged in.

NOTE 1: Unix distinguishes lower and upper case letters.

NOTE 2: You must use your physics user-name and password.

Everyone entering McGill is given a DAS account. But to access physics computers, you must have a physics account. If you don't have a physics account, ask your supervisor to get you one.

If you don't have a desktop set up on your desk yet, there is a common computer room on the second floor, room 230. To get in, press 1,3,5, in that order. There are also few Windows machines in that room as well as a couple of scanners.

2 You are In

Summary of this section

- `yppasswd` : Changes your password.

  Usage: To change your password for all the machines in the network,

  ```
  yppasswd
  ```

- `ls` : Lists files and subdirectories in the present directory.

  Usage : To list files and subdirs in the present directory,

  ```
  ls
  ```

  To list files and subdirs in the directory `sub_dir`,

  ```
  ls sub_dir
  ```

  To see if you have that file in the present directory,

  ```
  ls file
  ```

- `mkdir` : To make a directory

  Usage: `mkdir new_dir_name`

- `cd` : "change directory"

  Usage : To change to a subdirectory,

  `cd sub_dir`

  To go one up to the parent directory,

  `cd ..`

  To go to the home directory (the one you get when you login),

  `cd`

- `pwd` : "print working directory"

  Usage : To find out where you are,

  `pwd`

When you login, KDE window manager will come up. It may take a few seconds. The organization of KDE is very similar to Windows. The first thing you need to click on is the big 'K' icon at the bottom left corner. This is like the 'Start' button on Windows. Click it and you will see lots things such as applications, settings and logout.

Now unlike Windows, *Unix is all about command line.* So sooner or later, you have to know how to do things at the command prompt. The sooner, the better. The first thing to do is to find the 'xterm' icon from the bottom bar. It looks like a little terminal. Click it and you should see a window popping up with something like this:

```
HAL9000>
```

Now, $\boxed{\text{if this is your first login, you \textit{must} change your password.}}$

You can do that by typing

```
HAL9000> yppasswd
```

at the prompt. You will be asked to input your new password twice. Memorize your password. If you don't know, or you have forgotten your password, go to your system operator, beg for mercy, get a new *temporary* password, then change it to a new one.

Now that you are in, first let's see what you have. Type

```
HAL9000> ls
```

This results in something like

```
foo.txt         foo.tex         foo         foodir
```

You may think of `ls` as a shorthand for "list" though I have no idea if that was really the origin of the name. Unix command names tend to have peculiar origins, like `grep`[2]. Some are easy to memorize, for instance `cp` for "copy", but some names may seem to have no connection whatsoever to their functions (what do you think `awk` does?). But that's life. You win some, you lose some.

So, you have two objects `foo.txt`, `foo.tex` that look like files and and things called `foo` and `foodir`. To find more about each object do

```
HAL9000> ls -l
```

If the very first letter of an entry is `d`, that means that entry is a directory. If you see `x`'s in the first column, that means the object in principle can be run as a program.

To see the content of a subdir, type

```
HAL9000> ls foodir
```

If it is a subdir, then it will result in

```
oof.txt         oof.tex         oof         oofdir
```

[2] The original UNIX text editor "ed" has a construct g/re/p, where "re" stands for a regular expression, to Globally search for matches to the Regular Expression and Print the lines containing them. This was so often used that it was packaged up into its own command, thus named "grep". According to Dennis Ritchie, this is the true origin of the command.

If it is an ordinary file after all, it will just say

```
foodir
```

To see *all* the files including files that start with `.` (these are usually config files like `.tcshrc`),

```
HAL9000> ls -a
```

To find out more about `ls`, type

```
HAL9000> info ls
```

`info` is the command that retrieves information on the command you are interested in.

To read a file, type

```
HAL9000> less foo.txt
```

Other file readers include `more` and the primitive `cat`.

It is in general much better to organize your materials into subdirectories. To make a subdirectory,

```
HAL9000> mkdir new_dir_name
```

To change to a subdirectory, type

```
HAL9000> cd foodir
```

You can think of `cd` as an acronym for "change directory". Going one ladder up in the directory structure (to the parent directory) is

```
HAL9000> cd ..
```

NOTE: The present directory in Unix is always represented by a dot. That is, if you do

```
HAL9000> cd .
```

it does nothing because you are changing directory to where you are now.

To go to the home directory (the one you get when you login), simply type

```
HAL9000> cd
```

To find out in which corner of your subdirectory maze you are currently stuck in, type

```
HAL9000> pwd
```

`pwd` stands for ‘print working directory’. This will result in something like

```
/homes/hal9000/dave
```

The paths before you user name indicate on which disk your home directory is located. But be warned that sometimes `pwd` produces inconsistent result when you are logging in from someone else’s machine.

If you want to quit, type `logout` or `exit`. This will normally quit the x-term you are currently in. If you want to quit the session and get back to the login window, find a menu item under the ‘K’ button that says `logout`.

3 Read Files, Write Files, Copy Files, Move Files

Summary of this section

- `more` or `less` : To read a file. Use `less`.

  Usage :

  ```
  less file
  ```

- `nedit`, `kdedit`, `vi`, `emacs` : To edit a file.

  ```
  Usage : nedit file
  ```

- `cp` : To copy a file.

  Usage :

  ```
  cp original copy
  ```

  copies `original` to `copy`. If `original` is a file but `copy` is a directory, a copy of `original` with the same name will be made in that directory.

- `rm` : To remove a file.

  Usage : To remove a file

  ```
  rm file
  ```

  To remove all files excluding subdirectories (Be careful with this!)

  ```
  rm *
  ```

- `mv` : To move a file or a directory

  Usage :

  ```
  mv old_name new_name
  ```

  renames `original` to `copy`. If `original` is a file but `copy` is a directory, `original` will be moved to that directory. You can move directories around in exactly the same fashion,

- `rmdir` : To remove an *empty* directory.

  Usage :

  ```
  rmdir sub_dir
  ```

As I told you, you can use `more` or `less` to read a file. When using `more` or `less`, the space bar usually takes you one page further, pressing `b` will get you to the previous page, and you can search for a word by doing

```
[contents of a file displayed]

:/foo
```

Here the colon means the command line (It's given at the bottom. If you see it, you don't need to type that. If you don't, it doesn't hurt to type that.), the slash means search forward for the word 'foo'. To search backward, use `?` instead of `/`. To search the same word again, just press `n` for 'next'. To get out, press `q` for 'quit'.

To write a file, you need an editor. Under the 'K' button, you should see a menu item labeled 'editors'. There are many choices. It is up to you and your taste to settle on a particular one. Perhaps the easiest one is `nedit` or `kdedit`. But for anything that is more complicated than simple text, you need an advanced editor like `vi` or `emacs`. Try them out. At the command prompt, just say

```
HAL9000> nedit file
```

to edit `file` (new or old).

There is a neat feature in the Unix X-windows: **Two-click copy and paste**. If you want to copy *any text* that's displayed in *any windows* (except the previewers), put the cursor at the beginning of the text, press and hold the left mouse button and scan the text. This highlights the scanned text. Now release the left mouse, move the cursor to the position where you want a copy to be pasted and then click the middle mouse button. That's it.

To remove a file, do

```
HAL9000> rm foo.txt
```

To remove a directory, first `rm` all files in that directory. This can be accomplished by going into the directory, say, `foo_dir`

```
HAL9000> cd foo_dir
```

and issuing the command

```
HAL9000> rm *
```

The star `*` in Unix means "anything" or the "wild card". For instance, if you want to remove *all* files that start with `corrupted` and end with `.txt`, you can do

```
rm corrupt*.txt
```

However, this will not remove any subdirectory of `foo_dir`. In such cases, you have to repeat these steps.

Now go up to the parent directory,

```
HAL9000> cd ..
```

and issue the command

```
HAL9000> rmdir foo_dir
```

You can also do, from the parent directory of the one you want remove,

```
HAL9000> rm -r foo.dir
```

Here, `-r` means "recursively", and it will wipe out all the files and subdirectories under `foo_dir` including `foo_dir`.

$\boxed{\text{Be very careful using commands like } \texttt{rm *} \text{ or } \texttt{rm -r}.}$

If you do this in your home directory, you can easily wipe out all your work! And in Unix, unless there is a backup tape made (by the system operator and these are usually days old), **there is no way you can recover rm'ed files**. I repeat: *There is no way you can recover rm'ed files!* So always make copies of your important files.

To copy a file,

```
HAL9000> cp foo.txt goo.txt
```

This copies `foo.txt` to `goo.txt`. `foo.txt` is not destroyed. However, if `goo.txt` was an existing file, its old content is now *destroyed* and replaced by the new content. So, be careful.

To copy a directory, you should first make a directory

```
HAL9000> mkdir copy_dir
```

and copy the content

```
HAL9000> cp original_dir/* copy_dir/
```

This will copy all the files in `original_dir` to `copy_dir`. If you want to copy the directories as well, you can do

```
HAL9000> cp -r original_dir copy_dir
```

without first making `copy_dir`. The option `-r` here means 'recursively'. However, this is *dangerous*. If you have linked directories that happened to form a loop, the copying process will not end until it fills up the whole disk! If you are unsure, do the safe thing. Copy each directory manually.

To change a file name,

```
HAL9000> mv foo.txt goo.txt
```

This renames `foo.txt` to `goo.txt`. Again if `goo.txt` was an existing file, its content will be replaced. To move a file to another directory,

```
HAL9000> mv foo.txt foo_dir/
```

This moves `foo.txt` to the subdirectory `foo_dir`. If you now do

```
HAL9000> ls foo_dir
```

`foo.txt` will show up there. If you do,

```
HAL9000> mv foo.txt foo_dir/goo.txt
```

then this moves `foo.txt` to the subdirectory `foo_dir` and rename it to `goo.txt`. You can move directories around in exactly the same way.

4 Compile and Run

Summary of this section

- `gcc` or `cc` : C Compiler.

  Usage : To compile a C program,

  ```
  gcc -o prog_name file.c files_*.c -lm
  ```

- `g77` or `f77` : Fortran 77 Compiler.

  Usage : To compile a Fortran program

  ```
  g77 -o prog_name file.f files_*.f
  ```

- To run the compiled program

```
prog_name
```

To run the compiled program in the background

```
prog_name &
```

- Use the batch que to run a long job (taking more than 30 minutes).
- `<ctrl>c` : To interrupt a running program

  Usage : Press the `Ctrl` key and the `c` key together if you want to stop the running program
- `<ctrl>z` : To suspend a running program

  Usage : Press the `Ctrl` key and the `z` key together if you to want suspend the running program
- `bg` : To run the suspended job in the background, that is, the program is running but you get your command prompt back
- `fg` : To run the suspended job in the foreground, that is, the program is running and you don't get your command prompt back. Also to bring a process running in the background to the foreground.
- `jobs` : To list your jobs running in the background
- `ps` : To list all processes running in the background and foreground.

  Usage:

  To list processes started *in that shell only*,

  ```
  ps
  ```

  To list all your processes

  ```
  ps -u
  ```
- `kill` : To kill a job or a process

  Usage: First do

  ```
  jobs
  ```

to get the job number or do

```
ps -u
```

to get the process number. Then

```
kill %3
```

to kill the job number 3, for instance, or

```
kill process_number
```

to kill a process with `process_number`.

To overkill,

```
kill -9 %job_number
or
kill -9 process_number
```

Suppose you have written a C program file called `solve_everything.c`. To compile,

```
HAL9000> cc solve_everything.c
```

Here `cc` is the "C Compiler". In Linux this is just linked to the 'gnu cc' or `gcc`. So might as well use `gcc` in place of `cc`. Your choice.

If `solve_everything.c` is a simple program that doesn't require any library files, this will produce an executable file called `a.out`. You might want to check by typing `ls` to see if it is really there. If it is there, just type

```
HAL9000> a.out
```

and all the world's problem will be solved.

However, C programs are usually more complex and you don't want to use `a.out` for the name of every program you write. Also, you are going to need the library functions. The most important of all libraries, at least for me, is the math library. If you use any math in your program, that is, if there is a line in your program that says,

```
#include <math.h>
```

you need this library. Also, large programs usually have many separate `.c` files. So if you want to call your program `solve_this`, and if you need the math library, do

```
HAL9000> gcc -o solve_this solve_everything.c others_*.c -lm
```

The `-o` here means "output" and `-lm` means "link math library". The name of the library file used in this case is `libm.so` or `libm.a`. If the library you need is called `libWhatever.so` or `libWhatever.a`, then use `-lWhatever`.

Now you can run this program by typing

```
HAL9000> solve_this
```

Doing so makes you lose the command prompt while the program is running. To get the prompt back with the job running in the background, do

```
HAL9000> solve_this &
```

For Fortran programs, use

```
HAL9000> g77 -o solve_this solve_everything.f other_files.f
```

You can also use `f77` in place of `g77`, but that's just another name for `g77` anyway. To run it

```
HAL9000> solve_this

or

HAL9000> solve_this &
```

as before. Note that you don't need to specify a library for math functions in Fortran.

If something goes wrong and you want to stop the program running *in the foreground*, the interrupt signal in Unix is `<ctrl>c` which kills the program, or `<ctrl>z` which merely suspends the program. There are two ways to resume the suspended program. Sometimes you want to run the program in the background. That is, you want the program to resume but also want the command prompt back. No problem. Just say

```
HAL9000> bg
```

If there are many suspended runs, do first

```
HAL9000> jobs
```

to get the job number and do

```
HALL9000> bg %3
```

to restart, for instance, the job #3.

On the other hand, sometimes you want the program to resume in the foreground. That is, you don't want your command prompt back. In that case, just say

```
HAL9000> fg
```

`or`

```
HAL9000> fg %3
```

This job can be interrupted by `<ctrl>c` and `<ctrl>z` again.

Sometimes, your program may completely freeze a window while running in the foreground. In that case, you need to know the process number to kill it. To list processes started *in that shell only* (in practice, this means processes started in that xterm),

```
ps
```

To list all your processes

```
ps -u
```

If your job can take longer than 30 minutes, you should put it in the batch que: The format is as follows. To run a long job on `hal9000`

```
qsub -M -tjob_name hall9000 batch_script
```

For details, do

```
info qsub
```

If you run a program which does not give you a command prompt, it will be shut off after 30 minutes *unless* it is submitted using `qsub`.

You can also check the status of machines and jobs by doing

```
qstat
or
qstat machine_name
```

To kill a process or a background job, first do

```
jobs
```

to get the job number or do

```
ps -u
```

to get the process number. Then

```
kill %3
```

to kill the job number 3, for instance, or

```
kill process_number
```

to kill a process associated with `process_number`. Sometimes the process cannot be killed by simple `kill`. Then you have force it. To overkill,

```
kill -9 %job_number
```

```
or
```

```
kill -9 process_number
```

5 LaTeX, PostScript, PDF, Printing

Summary of this section

- `latex` : To LaTeX a LaTeXfile.

  Usage : to LaTeX `file.tex`

```
latex file.tex
```

- `xdvi` : To preview a `.dvi` file on an X-terminal.

  Usage : to preview `file.dvi`

  ```
  xdvi file.dvi
  ```

- `dvips` : To make a PostScript (ps) file to preview or to print.

  Usage : To make `file.ps` from `file.dvi`

  ```
  dvips file.dvi
  ```

- `ps2pdf` : To make a pdf file from a ps file.

  Usage : To make `file.pdf` from `file.ps`

  ```
  ps2pdf file.ps
  ```

- `gv`: To preview a PostScript file or a pdf file.

  Usage : to view `file.ps`

  ```
  gv file.ps
  ```

- `acroread  or  xpdf` : To preview and print a pdf file.

  Usage : to view `file.pdf`

  ```
  acroread file.pdf
  ```

  To print, click `file` then `print` in `acroread`.

- `lpr` : To print a PostScript file or a text file.

  If your file is not a ps file or an ascii file, you *must* convert it to PostScript or ascii before sending it to a printer.

  Usage : To print a PS file on the default printer

  ```
  lpr file.ps
  ```

To print a PS file on the printer named `halprt`

```
lpr -Phalprt file.ps
```

To print an ascii file on the default printer

```
lpr file.txt
```

- `lpstat` or `lpq`: To ask a printer how it is doing.

  Usage : To ask the default printer

  ```
  lpstat
  ```

  To ask `halprt`

  ```
  lpstat -phalprt
  or
  lpq -Phalprt
  ```

  Note `-p` for `lpstat` and `-P` for `lpq`.

- `lprm` : To remove a printing job. This is currently not implemented.

So, you have written a LaTeXfile, it might say something like

```
\documentclass[12pt]{article}
\begin{document}

I won't do that if I were you, Dave, 'cause
\begin{equation}
{\bf F} = m{\bf a}
\end{equation}

\end{document}
```

Let's call it `hal.tex`.

To LaTeX it, just type

```
HAL9000> latex hal.tex
```

You can also omit the `.tex` extension

```
HAL9000> latex hal
```

If you didn't make any mistake, it will produce `hal.dvi, hal.aux, hal.log`. Use `ls` to see everything is there. If they are, that's it. You've done it.

As the name indicates, `xdvi` reads the `.dvi` file you produced and shows it on the X-terminal screen, so be sure that you have the `.dvi` file, in our case, `hal.dvi`, before you do

```
HAL9000> xdvi hal
```

If all goes well, now a window will pop up, showing you what your document will looks like when printed:

I won't do that if I were you, Dave, 'cause

$$\mathbf{F} = m\mathbf{a} \tag{1}$$

To print it, you need to make a PostScript file first. Making the PostScript file is usually done by a program called `dvips`. Just type

```
HAL9000> dvips hal.dvi
```

This will produce `hal.ps` from the `hal.dvi` file. To preview this file,

```
HAL9000> gv hal.ps
```

If you want to produce a pdf file, then instead do

```
HAL9000> dvips -Ppdf hal.dvi
```

This will produce `hal.ps` that's suitable for pdf conversion. To convert,

```
HAL9000> ps2pdf hal.ps
```

That produces `hal.pdf`. To preview a pdf file,

```
HAL9000> acroread hal.pdf
```

If acroread is not installed on your machine use `xpdf` or `gv`.

If all went well, now you would want to print the thing and admire what you have done. The usual printing command in Unix is `lpr` (that's not "laser printer", that's "line printer"). So

```
HAL9000> lpr hal.ps
```

will usually print the thing out of the default printer.

If you want to print on a printer other than the default one, you have to know the name of the printer you want to use. Ask your system operator. To use a particular printer, say, `halprt`, type

```
HAL9000> lpr -Phalprt hal.ps
```

To check if indeed the printer is functioning and your thing is being printed out, or if there are a lot of loads on that printer, type

```
HAL9000> lpstat -phalprt
```

This will give you job-name, user-name, job-number, size and status. Exact names and order of things being reported may vary slightly from system to system. If this does not work, use instead

```
HAL9000> lpq -Phalprt
```

Note that here it's the upper case `-P` where in the case of `lpstat` it is the lower case `-p`.

The command `lpr` can take a lot of options such as turning on and off two-sided printing, multiple copies, manual input, etc. Options for each printer is posted on the cover of each printer. For more details, go to `http://www.physics.mcgill.ca/~roderick/printers.html`

Now I bring you these important messages from our system operator:

Color Printers

```
 Words of caution (for Tektronix color printers):
  . Send ONLY text or POSTSCRIPT (use your favorite converter)
  . NEVER turn the unit off
  . Use ONLY Tektronix SPECIFIC transparency paper (we can help you
        get some)
  . NEVER ;( .. NEVER remove the paper tray UNLESS it is requested from
    the printers display panel.
```

Another CLARIFICATION for printer ABUSERS..

```
We have an increasing number of complaints about people abusing
their printing privileges. Please remember, full color pages and
printing transparencies have a much higher cost then black and white
pages.

The printers and maintenance are paid by RESEARCH funds and these funds are
not for personal expenses. Admittedly, many of us use the printers for
'private' (non-research) from time to time, but some behaviour is
CLEARLY shocking not to say inadmissible.
Yesterday's printing of a Quebec tax form, including users' guide, probably
about 100 pages, while available in many locations around the city, is
a case in point.

Be aware, we have printer accounting running and we know which login
account prints where. Printing privileges can be revoked.

Also the printers are a shared resource, so be considerate.  Think
before submitting print jobs, restrict LONG jobs to less active hours
or less used printers.

Finally, PLEASE .. PICK UP YOUR ** JOBS **. If it's not worth picking
up, it is not worth printing.
```

6 E-Mails, Sending and Receiving Files, Tar, Gzip

Summary of this section

- `pine` : To read and send e-mails.

  Usage :

  ```
  pine
  ```

- `mail person@some.where.com < file` : To send a file without going through an e-mail program.

- `tar` : To collect many files and directories in one file preserving the directory structure.

  Usage : To make a `tar` file `all_files.tar`

  ```
   tar cvf all_files.tar file1 file2 files*
  ```

  To un-tar `all_files.tar`, go to the directory to unfold and do

  ```
  tar xvf all_files.tar
  ```

- `gzip`, `bzip2` or `compress` : To compress a big file.

  Usage : To compress or zip a file

  ```
  gzip file
  or
  bzip2 file
  ```

  To decompress, or to unzip a zipped file

  ```
  gunzip file.gz
  or
  bunzip2 file.bz2
  ```

Now that you have created a wonderful program or a beautiful document, maybe you'd like to show it to somebody or maybe somebody already sent you their handy work. So to read e-mails, type

```
HAL9000> pine
```

That'll get you into `pine`. I can't explain all the intricacies of `pine` here, but it's quite intuitive. The cursor control usually works as one expects it to work, and there is a menu bar at the bottom showing what the available commands are and what they are for. To summarize:

- To read an email, just `enter`.
- To 'compose' a new email, press `c`.
- To 'reply', press `r`.
- To 'delete', press `d`.
- To 'save', press `s`.
- To change levels, use `<, >`. Try it. See what it does.
- To get out of `pine`, press `q`.
- To send files, use 'attachment'.
- To change the editor, go to the top level where you see something like

  ```
    PINE 4.58   MAIN MENU  Folder: INBOX  10 Messages

     ?     HELP                -  Get help using Pine
     C     COMPOSE MESSAGE     -  Compose and send a message
     I     MESSAGE INDEX       -  View messages in current folder
     L     FOLDER LIST         -  Select a folder to view
     A     ADDRESS BOOK        -  Update address book
     S     SETUP               -  Configure Pine Options
     Q     QUIT                -  Leave the Pine program

     Copyright 1989-2003.  PINE is a trademark of the University of Washington.
  ? Help                     P PrevCmd                 R RelNotes
  O OTHER CMDS > [ListFldrs] N NextCmd                 K KBLock
  ```

  Go to `SETUP` then choose `CONFIG`. In it you will find many, many options including the editor.

To send a file directly to other user without going through an e-mail program, you can do

```
HAL9000> mail person@some.where.com < foo.txt
```

But this works *only if* `foo.txt` is an ascii file. To send non-ascii files, use attachment as explained below.

Now suppose you want to send files `foo_1.txt` through `foo_20.txt`. You can send it one by one. However, that's time consuming, prone to errors, and quite frankly, the other person will get horrified to receive 20 separate mail messages for 20 separate files because he/she has to edit out all the mail headers one by one, put them in the right directories, etc., the horrors. So, be smart and considerate and use `tar`.

The Unix utility `tar` comes in quite handy when you want to collect a number of files, even directories, in one file. The word `tar` originally meant "tape archive" but these days, it is widely used to just collect a lot of files in one convenient file. To do so

```
HAL9000> tar cvf all_foo.tar file_1 file_2 foo_*.txt
```

Here `cvf` means "create `(c)` a file `(f)` showing what you are doing verbatim `(v)`". As before, `*` in Unix means "anything" or the "wild card". This will create a file named `all_foo.tar` which contains `file_1`, `file_2` and `foo_1.txt, foo_2.txt, ..., foo_a.txt, foo_b.txt, ...`. So now send `all_foo.tar` to the person at the other end.

A `tar`'ed file is a binary file. And as such, it loses bits and pieces if you send it through e-mail as a *body text*. Send it using 'attachment'. When you compose a message, you will see 'Attachment' in the header. Just provide the path to the file you want to attach.

A word of caution. If you make a mistake and omit the `tar` file to be made (`all_foo.tar` in the example above), then the first file listed will be taken to be the `tar` file to be made (`file_1` in the example above). You can guess what will happen then: The content of that first file is destroyed. So be careful.

Now, if you happened to be the person at the receiving end of a `tar` file, do

```
HAL9000> tar xvf all_foo.tar
```

Here `xvf` means "extract `(x)` from a file `(f)`, showing what you are doing verbatim `(v)`". This will now create all the `foo_*.tex` files in the directory `all_foo.tar` is in.

If you received a `tar.gz` file, you can first `gunzip` it and do the above or you can simply do

```
HAL9000> tar zxvf all_foo.tar.gz
```

Note the additional `z`.

If the files you are sending out are really big ones, then you might wish to compress it first. The original Unix compress command is, funnily enough, `compress`. which produces a `.Z` file. These days, the "GNU zip", that is `gzip` or the newer `bzip2` are more commonly used since they are more efficient. To `gzip` or `bzip2`,

```
HAL9000> gzip all_foo.tar
or
HAL9000> bzip2 all_foo.tar
```

This produces a file `all_foo.tar.gz` (`all_foo.tar.bz2`) where the extension `.gz` (`.bz2`) means it is `gzip`'ed (`bzip2`'ed). To decompress,

```
HAL9000> gunzip all_foo.tar.gz
or
HAL9000> bunzip2 all_foo.tar.bz2
```

Incidentally, `gunzip` will also unzip the Zipped files with `.Z` extension.

7 Getting on to the Web

Summary of this section

- `mozilla &` : To run Mozilla in the background and get on to the WWW.

- `netscape &` : To run Netscape in the background and get on to the WWW.

- `~/WWW` : McGill homepage location for HEP/TNP/ASTRO.
- `/WWW/user_name` : McGill homepage location for CMP/GANG.

Getting on to the World Wide Web is as easy as typing

```
HAL9000> mozilla &
```

Mozilla is better than Netscape in my opinion. The `&` sign at the end is the "background" sign. Mozilla will now run in the "background" mode.

You also can have a personal webpage set up. If you beling to HEP/TNP/ASTRO, create `WWW` subdir from your home directory and put `html` files there. The file named `index.html` in this directory is your home page. If you beling to CMP/GANG, the put `html` files in `/WWW/user_name` instead.

The homepages are is updated daily. Please do not abuse the disk space.

8 Connecting to/from Outside

- `ssh` : To securely login to other computers

  Usage :

  ```
  ssh user_name@other.machine.some.where
  or
  ssh -X user_name@other.machine.some.where
  ```

  The option `-X` lets in the X signals. However, unless the machine is local, this is slow.

- `scp` : To securely copy a file from one machine to another.

  Usage : To copy `file_1` from the current machine to another and name it as `file_2`

  ```
  scp file_1 user_name@other.machine.some.where:file_2
  ```

  To copy from `file_1` from another to the current machine and rename it as `file_2`

```
scp user_name@other.machine.some.where:file_1 file_2
```

- `sftp` : To securely transfer file from one account another

  Usage : `sftp user_name@other.machine.some.where`

- To get your e-mails from outside McGill: Log in using `ssh` and do `pine`.
  If you want web-based email service, configure your browser/outlook/thunderbird with IMAP with SSL and set the Incoming server to

  `imap.physics.mcgill.ca`, SSL, port 993.

  Outgoing server to

  `mx1.hep.physics.mcgill.ca`, TSL, port 587.

`Ssh` is 'secure shell'. It encrypts communication between two computers to prevent interception of information. If you do not have `ssh` in your home machine, you can get a copy from Paul (our system manager) or download it from `www.openssh.com`. The package also includes `scp` which acts like `cp` but secure, and `sftp` which acts like `ftp` but secure. If you use Windows at home, install `PuTTY` from `http://www.chiark.greenend.org.uk/~sgtatham/putty/` or `WinSCP` from `http://winscp.sourceforge.net/eng/`.

To read mails from home using your browser:
You must know your current physics password in order to be able to read mail. If you don't know your physics password, you will have to contact the operators.

You can use your browser or favorite mail reader by making the following configuration changes:

- Incoming Mail Server: `imap.physics.mcgill.ca`. Set the port number to 993 and secure connection type SSL.

- Outgoing Mail (SMTP) Server: the SMTP server of your ISP. Physics SMTP server is `mx1.hep.physics.mcgill.ca`. You may need the port number (587) to connect from home.

- Your identity (name, email address, username, etc.)

- Do not turn on Secure authentication.

9 Getting More Information

Summary of this section

- `info` : To read info pages.

  Usage : to find out about a command

  ```
  info command
  ```

  To search info pages for a keyword

  ```
  info --apropos=keyword
  ```

- Fire up Mozilla. Go to `http://www.google.com` and search for 'Unix tutorial'.

The best way to get more information is to use the `info` command. You might want to try

```
HAL9000> info info
```

first to find out how to use `info` effectively.

Perhaps you heard about a command called `awk` but have no idea what it does. To find out, type

```
HAL9000> info awk
```

This will show you the info pages for the command `awk` which shows what it does, how to use it, and where to find more info. Now, suppose you want to do a certain thing, but don't quite know if there is a command for it. For instance, suppose you want to check your spelling, but don't know if there is a program for it. To find out, you can search info pages for the key word, for instance, you may do

```
HAL9000> info --apropos spell
```

Also, it is an excellent idea to fire up a Mozilla, go to `http://www.google.com` and search for `Unix tutorial`. You should get a ton of hits.

10 Other useful Commands and Programs

- `ypchsh` : To switch your shell globally. Default is `tcsh`.
- `chmod` : To change the attribute of a file or a directory.

  Usage: `chmod [ugoa][+-][rwx] file`

  For instance if you don't want others to read your file,

  ```
  chmod o-r your_file
  ```

  That is, "change the mode of `your_file` so that others can't (`-`) read it".

  `u` is the user, that's you. `a` is all and `g` is group. `+` adds the attributes, `-` subtracts them, `w` is write privilege, `x` is execute privilege.

- To check filesystem disk space usage

  ```
  df
  ```

- To check your own disk space usage

  ```
  du
  ```

- To add a directory to the command path : Edit the `.tcshrc` file in your home dir. For instance, add

  ```
   set path = (. $path)
  ```

  if you want to put the current directory before the default path or

  ```
   set path = ($path .)
  ```

  to put it after the default path.

- To convert one graphics format to another

  ```
   convert picture.eps picture.jpg
  ```

This converts `picture.eps` to `picture.jpg`. The known formats are `.ps, .eps, .pdf, .png, .pnm, .gif, .jpg, .tiff`, etc. If `convert` is not installed on your machine, do

```
which pngtopnm
```

and note in which directory `pngtopnm` is in. Go there and `ls`. You should see 200+ converters for almost all combinations.

- To convert a ps file to an epsi file

  ```
  ps2epsi picture.ps
  ```

  This produces `picture.epsi`.

- To convert a ps file to an ascii file

  ```
  ps2ascii paper.ps output.txt
  ```

  This extracts text from `paper.ps`.

- To view a graphics file

  ```
  display picture.jpg
  ```

  If `display` is not installed, then use `xv`.

- `nice` : To be nice to other people using the same machine

  ```
  nice prog_name
  ```

  so that your program does not hog all the resources.

- `>` and `<` : Redirect. If a program takes standard input (STDIN) from the keyboard, you can create a file with all the data and do

  ```
  prog < input_dat
  ```

  and the program will take the lines in `input_dat` as if it is given at the command prompt followed by `return`.

  If the program prints out to the standard output (STDOUT) or on your screen, you can redirect it to a file using `> output_data`. Namely

```
prog < input_data > output_data
```

Note that redirect destroys `output_data`. If you want to append to the file use `>> output_data`.

- Pipe `|` : Pipe is used when you want to string programs that take a *file* as an input but writes on to STDOUT. Take a look at this example. Suppose you have a jpeg file you want to convert to a png file. And suppose your `convert` does not function somehow. Now there is this command

  ```
  jpegtopnm picture.jpg
  ```

  which writes out a pnm file to STDOUT, or *on your screen.* Don't try this. It'll mess up your screen. You can always redirect it

  ```
  jpegtopnm picture.jpg > picture.pnm
  ```

  then use

  ```
  pnmtopng picture.pnm > picture.png
  ```

  again redirecting it to a file. This process, however, can be short-circuited using pipe:

  ```
  jpegtopnm picture.jpg | pnmtopng > picture.png
  ```

  This is also very useful when a program produces a massive amount of output that just flies by. In that case try

  ```
  prog_name | less
  ```

  You can then treat the output of this program as if it is a file and use `less` to navigate around.

- `grep` : To globally search regular expression and print the line. Or in English, grabs line that contains a word or phrase from a file or files and writes to STDOUT.

```
grep word file.txt
or
grep "A word with you, sir." file*.txt
```