UNIX basics

The UNIX operating system was born in the late sixties and through the years different versions and flavors have appeared. UNIX is typically used as a text-based operating system and, following the original UNIX philosophy, the commands are often “shortcuts“ and include options that have to be remembered. These facts are the main hurdles for UNIX beginners, as today everybody is used to working with graphical user interfaces, mouse clicking, drag-and-drop etc. Another fact that gives many beginners a hard time is that there are often several ways (commands, command sequences) to do something. But once you know only a few UNIX commands you are in business! And with increasing experience you realize that UNIX is very consistent and powerful.

General notes

The following tutorial is based on the tcsh shell, currently installed on “Helix” (the SUN multiprocessor server) and the examples are given for the user "szoller" with home directory /home/szoller. Don't worry at this moment if do not know what a shell is or a home directory.

Remember that UNIX is **case sensitive**. Be sure to write the password, commands, options and file names with the correct spelling. When UNIX is not doing what I want, I often realize that I misspelled a command or a filename. So stay calm and try again! UNIX has a built in **manual**, which explains all the commands and the available options. If you need information on the manual itself type man man (and press return). If you need help on, let's say the find command, type man find. Unfortunately, the manual is somewhat cryptic and the examples are sometimes hard to understand. Nevertheless it can be very useful.

A nice feature of the UNIX command line is its **command history**. By pressing the up and down arrow key you can "walk" through the last commands you have actually used. This can increase the speed of writing commands tremendously, especially when you have to do some repetitive tasks.

Because the communication with Helix is text-based, it might happen that the system does not respond or you typed the wrong command and your window "freezes". In this case try the **suspend sequence** control-Z (press both keys simultaneously). If this control sequence does not help, try control-C, the **kill command**. This will "kill" the running job (caution: data may be lost, depending on the command you were running). See the chapter on job control for more information on suspend and kill.

Please do not just quit your NiftyTelnet (Mac) or Putty (PC) application, but first use the **exit command** to finish the Helix session and then you can quit.

If you want to know more about UNIX and the commands, then you might consider buying a book or check out some web pages. A few **books** I can recommend are:

- Introduction to UNIX, D.I. Schwartz, Prentice Hall, 1999, 133 pp. ca. $ 20.- . This is very readable introduction for beginners with lots of examples.
- Learning the UNIX Operating System, J. Peek et al., O'Reilly, 1998, 93 pp., ca. $ 13.- . A good introduction, less "example driven" than the first one.
- UNIX System V, M.G. Sobell, Addison-Wesley, 1996, 831 pp., ca. $ 38.- . A comprehensive introduction and reference guide.
- UNIX in a Nutshell, A. Robbins, O'Reilly, 1999, 598 pp., ca. $ 25.- . This is a dry quick reference, explaining all the commands and options. For beginners probably sometimes too "quick".
- UNIX Power Tools, J. Peek et al., 1997, 1073 pp., ca. $60.- . The UNIX Bible for the serious UNIX user (and programmer) with lots of hints from experts.

Two useful **internet pages** are the "UNIXhelp for Users": http://dapsas1.weizmann.ac.il/UNIXhelp/Pages/
and the "Introduction to UNIX": http://www.mhpcc.edu/training/vitecbids/UnixIntro/UnixIntro.html

Files and directories: make new directories, copy, move and delete files

We assume that user szoller has successfully logged-in and uploaded the two text files example1 and example2 (check out the other tutorials available on http://hpcc.fmnh.org if you do not know how to do this). A “Helix” window opened and shows system information and the "prompt" helix /home/szoller> UNIX is now waiting for a command (compare Figure 3 in the tutorial "Start a Helix Session").

ls
list
files

1) First we would like to know the files and subdirectories in the home directory. Type **ls** (for list) and hit return:

```
helix /home/szoller> ls
helix /home/szoller> Bambe1    Paup1    Paup2     its.nex     example1     example2
```

The helix window shows the content of the szoller directory. The file system is hierarchical and has a tree structure. The first three entries shown are subdirectories (I know that because I created them) and the others are simply files. It is good practice to have directory names begin with an uppercase letter.

mkdir
make
directory

2) Now lets make a new subdirectory, called Bambe2. Type **mkdir** Bambe2 (for make directory). Do not forget to put a space between mkdir and Bambe2.

```
helix /home/szoller> mkdir Bambe2
helix /home/szoller> Bambe1    Bambe2   Paup1    Paup2     its.nex     example1     example2
```

A new subdirectory has appeared. Now copy the file example1 into that directory. **Caution**: UNIX does not warn you when a command you give will overwrite some existing file! The overwritten file is lost and cannot be recovered!

cp
copy
files

3) Type the copy command (**cp**) shown below. This copies the file example1 into the specified directory. Be sure to put a space between example1 and /home...

```
helix /home/szoller> cp example1 /home/szoller/Bambe2
```

A shorter version would be:

```
helix /home/szoller> cp example1 Bambe2
```

this works, because Bambe2 is a subdirectory of the current working directory (/home/szoller/).

mv
move
files or
rename

4) To move a file into a directory (like dragging on the Mac or PC) use the move command **mv**:

```
helix /home/szoller> mv example2 Bambe2
```

typing **ls** shows that example2 is no longer in the home directory. The file example1 is still there because we put a copy in Bambe2, leaving the original in the home directory:

```
helix /home/szoller> ls
helix /home/szoller> Bambe1    Bambe2   Paup1    Paup2     its.nex     example1
```

Note: Move (**mv**) is also used to rename files. The command mv example2 example2.old would rename the file example2 to example2.old without moving it.

cd
change directory

5) Now lets go to that new subdirectory. Type **cd** Bambe2 (for change directory) and hit return:

```
helix /home/szoller> cd Bambe2
helix /home/szoller/Bambe2>
```

The prompt shows now that we are in directory Bambe2. If you type **ls** you will see the example1 and example2 files:

```
helix /home/szoller/Bambe2> ls
helix /home/szoller/Bambe2> example1   example2
```

Note: If you know the full path of a directory, you can go to it directly. If you are for example in the home directory (/home/szoller) and you know there is a subdirectory called /Run1 in the /Bambe1 directory, you can go there directly by typing cd /home/szoller/Bambe1/Run1

rm
remove files (delete)

6) We need only file example2 in this directory, so lets delete example1. Type the command **rm** (for remove). **Caution**: remove means gone forever on UNIX!

```
helix /home/szoller/Bambe2> rm example1
```

type ls to see the files in Bambe2:

```
helix /home/szoller/Bambe2> ls
helix /home/szoller/Bambe2> example2
```

cd ..
change directory

7) To move back, one level higher in the directory hierarchy (to /home/szoller), just type cd **..** The double-dot stands for the parent directory. Each directory has a parent directory (and only one), similar to folders on your Mac or PC that are located within another folder.

```
helix /home/szoller/Bambe2> cd ..
helix /home/szoller>
```

find
search files

8) The command **find** lets you search for files. This command requires some options to be useful (-type for file type, -name for file name and -print to actually print the result). The syntax is somewhat strange but it works best as shown below. Do not forget the dot. This command looks in your current directory (represented by the dot after find) and all subdirectories for files with the name its.nex. It will list all the items found, including their paths (./its.nex below). A path is a string of directory names such as /home/szoller/Bambe2> that gives the exact location of a file.

```
helix /home/szoller> find . –type f –name 'its.nex' -print
./its.nex
helix /home/szoller>
```

some shortcuts

Note: Wherever you are in the directory tree, typing **cd** brings you back to your home directory. Assume we are in the /PrelimResults directory:

```
helix /home/szoller/Paup1/Run1/PrelimResults> cd
helix /home/szoller>
```

Note: the tilde ~ is a shortcut for the home directory (here /home/szoller). For example `cd /home/szoller/Bambe2` is equivalent to `cd ~/Bambe2`
If you want to go directly to some subdirectory of your home directory use ~ as shown below:

```
helix /home/szoller/Paup1/Run1/PrelimResults> cd ~/Bambe2
helix /home/szoller/Bambe2>
```

Creating and handling text files

Sometimes you need to write short text files from scratch.

cat
create a text file

1) An easy way to do this is by using the command **cat** with the sign for redirection (>). Type **cat** followed by > and then the name of the new file:

```
helix /home/szoller> cat > textfile1
```

You can now type the text, including carriage returns etc. but once you have changed to a new line you cannot go back to the previous one. When you are done, hit return, so that the cursors jumps to a new line and then press control-D (both keys simultaneously). Below is a "screenshot", I wrote the three lines "this is my . . . . . this is the last line" and then hit return and control-D :

```
helix /home/szoller/Bambe2> cat > textfile1
this is my textfile1 and this is line number 1
to jump to the next line I press return
to exit I press return and then control-D
this is the last line
helix /home/szoller/Bambe2>
```

A new file named textfile1 has appeared in the directory Bambe2 (you could check with **ls**).

more
take a look at a file

2) Use the command **more** to take a look at a file without changing it:

```
helix /home/szoller/Bambe2> more textfile1
```

If the file is longer than one window, the output stops. Hit the space bar to show the next window full of text. Hit the key b to go back one page. Hit the key q to quit the output and go back to the promt.

tail
take a look at a file's end

3) If the file is large it might be useful to have a look at the end of it. Use the **tail** command, which shows by default the last 10 lines. If you want to see the last two lines use the option (-2) as shown below:

```
helix /home/szoller/Bambe2> tail -2 textfile1
to exit I press return and then control-D
this is the last line
helix /home/szoller/Bambe2>
```

head
look at the file's beginning

4) The command **head** does the same job but for the beginning of the file:

```
helix /home/szoller/Bambe2> head -2 textfile1
this is my textfile1 and this is line number 1
to jump to the next line I press return
helix /home/szoller/Bambe2>
```

Note: If you have to do more sophisticated editing or need to know how to edit an existing file check out the tutorial "vi – a text editor"

Job control

UNIX allows you to run several processes (jobs) at the same time. Jobs can run in the foreground or in the background and you can suspend, restart and delete (kill) jobs.

suspend a job

1) First you need to have a job running. Let's make a text file named textfile2 with cat, but instead of finishing it with control-D we **suspend** it with control-Z:

```
helix /home/szoller/Bambe2> cat > textfile2
this is textfile2, line number 1
I suspend the process with control-Z
^Z
Suspended
helix /home/szoller/Bambe2>
```

bg
put a job in the background

2) Now type the command **bg** to put the suspended job in the background. There it will be automatically restarted unless it is a job that is waiting for input.

```
helix /home/szoller/Bambe2> bg
[1]    cat > textfile2 &
[1]  + Suspended (tty input)        cat > textfile2
helix /home/szoller/Bambe2>
```

jobs
show jobs

3) Now you could start some other jobs, write some other text files etc. To know the process (or job) number type **jobs** :

```
helix /home/szoller/Bambe2> jobs
[1]  + Suspended (tty input)        cat > textfile2
```

fg
bring a job back to the foreground

The cat job has the number 1. We can use this number with the **fg** command to bring the job back to the foreground:

```
helix /home/szoller/Bambe2> fg %1
cat > textfile2
```

ps -ef
show jobs

4) If you want to know what other processes are running, it is best to use the command **ps**. Try the option **–ef** to show more detailed information.

```
helix /home/szoller/Bambe2> ps –ef
```

In this form **ps** shows you the User ID (UID), the Process ID (PID), a unique number assigned to a process, the run time (TIME), the name of the command (CMD) and some less important information. Depending on the number of users and processes the list can be quite long. To sort out your own processes, use the option **-fu** and **your username**:

ps -fu
get matching iobs

```
helix /home/szoller/Bambe2> ps –fu szoller
  UID        PID  PPID  C    STIME    TTY      TIME  CMD
szoller   27016 26987  0 18:37:22  pts/0    0:00   cat
```

kill
erase a job

5) We now know the process ID of the **cat** command (PID# is 27016). By using the command **kill** in combination with the process ID we can "kill" the **cat** process:

```
helix /home/szoller/Bambe2> kill 27016
helix /home/szoller/Bambe2>
[1]    Terminated                 cat > textfile2
helix /home/szoller/Bambe2>
```

Check the processes (ps –uf szoller). The command cat is no longer on the list.

Note: If you have logged out and logged in again later, the job command will not show you the jobs of the previous terminal window (or: shell). In this case you have to use the command ps. If you have several jobs running it might be useful to write down the process IDs, just in case you have to kill a job later. You don't want to kill the wrong job ;-)