1 Demonstration 1: converting multiple image files

Problem: You are asked by a journal to submit all of your figures as jpg when you have all of them stored as pdf files.

Solution - the "Windows" way: open up an image manipulation program, then select each file individually and save each of them in the new format. Probably the fastest way if number of images is less than ten and you never have to do it again.

Solution - the "Unix" way: write a shell script to convert ALL pdf files in a directory. The script may take time to write, but after that the command can be run in an instant for ANY number of files.

1.1 Techniques Required

- navigate the filesystem
- execute commands from the terminal
- open up an editor
- write a for loop
- search and replace

1.2 Before We Start

These examples are using bash which is a particular type of command line language. The default configuration file has not been setup on the EPS computers so you need to create your own. I've found a pretty standard setup and put it on my website here. Simply download the file and save it in your home directory.

When you first open the terminal your command line will be placed in the home directory. In order to work with files in different directories, you need to be able to move around the filesystem. The first three commands you need are

- `cd` - short for change directory
- `ls` - short for list contents
- `pwd` - short for print working directory

Try out each of these three commands by typing them in a terminal and pressing return. See command list for more details.

WARNING do not copy/paste from pdf documents in plain txt files, it DOES NOT WORK! Try to type out the commands manually.

1.3 How to write the script

First we need some files to work with. Open up your web browser and download the zip file found here and extract in your home directory. Now open up a terminal and check that the extraction has worked in the correct way, issue the command

```
ls Unix
```

to see the contents of the directory. You should see the output

```
convertFiles-0.sh  page12-13.pdf  page16.pdf      page27.pdf    page8s.pdf
convertFiles-1.sh  page12-18.pdf  page17-18.pdf   page30.pdf
convertFiles-2.sh  page13-18.pdf  page23a-19.pdf  page5-19.pdf
page10-13.pdf      page14-13.pdf  page24.pdf      page6-13.pdf
```

Now navigate to the directory you have extracted with the command:

```
cd Unix
```

Next we are going to test the command to convert a pdf file to a jpg. The command we are going to use is `convert`, see the manual page by entering

```
man convert
```

Here the syntax is:
`command -option` *argument* `-anotherOption` *argument* `input output`
and we specify the following

```
convert -geometry 800x800 -density 200x200 -quality 100
           page10-13.pdf page10-13.jpg
```

After entering this command you should see the file `page10-13.jpg` appear in the directory, a jpg version of the file. Use the file browser to open the directory and look at the file.

Once we know how this command works we now need to write a loop to run it on each and every pdf file in the directory. To do this it is easier if we start writing our commands in a script. A script is simply a collection of commands that you would enter at the command line saved inside a text file, that can be run as a single command.

1.3.1 Creating the script

Use your file browser to open the "Unix" directory, double click on the file `convertFiles-0.sh`. An editor should now open. Inside the file you should see written

```
#!/bin/bash
#
# list all the files ending with a "pdf"
ls *.pdf
```

In order that the script is enabled to run we need to make it executable. Because it has been extracted from a zip file it should already be executable, you can check this by executing the command

```
ll convertFiles-0.sh
```

you should see something like this

```
-rwxr-xr-x 1 pjohnson users 65 Sep 16 11:40 convertFiles-0.sh
```

Note the `x`'s here mean that the program is executable. To make it executable (if it is not already) type

```
chmod +x convertFiles-0.sh
```

to add execute permission. If you now run the script from the command line like so

```
./convertFiles-0.sh
```

you will see the following output

```
page10-13.pdf  page13-18.pdf  page17-18.pdf   page27.pdf    page6-13.pdf
page12-13.pdf  page14-13.pdf  page23a-19.pdf  page30.pdf    page8s.pdf
page12-18.pdf  page16.pdf     page24.pdf      page5-19.pdf
```

All the script is doing is just running that single command.

1.3.2 Making a loop

The syntax on a for loop is as follows:

```
#!/bin/bash
#

for fileName in `ls *.pdf`
do

   # can now issue a command where ${fileName} will be one of the pdf files
   # output the fileName to screen
   echo ${fileName}

done
```

Enter the text as above in your script and run it. the output is

```
page10-13.pdf
page12-13.pdf
page12-18.pdf
page13-18.pdf
page14-13.pdf
page16.pdf
page17-18.pdf
page23a-19.pdf
page24.pdf
page27.pdf
page30.pdf
page5-19.pdf
page6-13.pdf
page8s.pdf
```

Here we can now issue a command on each file ending with pdf. For example, try creating a backup of each file, by adding the following command under the `echo` command.

```
cp -v ${fileName} ${fileName}_backup
```

Run the script again you should see

```
 page10-13.pdf
page10-13.pdf -> page10-13.pdf_backup
page12-13.pdf
page12-13.pdf -> page12-13.pdf_backup
page12-18.pdf
page12-18.pdf -> page12-18.pdf_backup
page13-18.pdf
page13-18.pdf -> page13-18.pdf_backup
page14-13.pdf
page14-13.pdf -> page14-13.pdf_backup
page16.pdf
page16.pdf -> page16.pdf_backup
page17-18.pdf
page17-18.pdf -> page17-18.pdf_backup
page23a-19.pdf
```

```
page23a-19.pdf -> page23a-19.pdf_backup
page24.pdf
page24.pdf -> page24.pdf_backup
page27.pdf
page27.pdf -> page27.pdf_backup
page30.pdf
page30.pdf -> page30.pdf_backup
page5-19.pdf
page5-19.pdf -> page5-19.pdf_backup
page6-13.pdf
page6-13.pdf -> page6-13.pdf_backup
page8s.pdf
page8s.pdf -> page8s.pdf_backup
```

The file `convertFiles-1.sh` contains the solution at this stage.

1.3.3 Changing file.pdf to file.jpg

This step is a little bit more complex and would normally require a websearch to find out how to do it. Try adding the following command after the copy command

```
echo ${fileName} | sed 's/\.pdf$/\.jpg/'
```

Run the script and you should see

```
 page10-13.pdf
page10-13.pdf -> page10-13.pdf_backup
page10-13.jpg
...
```

so `page10-13.pdf` has been changed to `page10-13.jpg`. This can now be used in the original command like this

```
convert -geometry 800x800 -density 200x200 -quality 100
        ${fileName} `echo ${fileName} | sed 's/\.pdf$/\.jpg/'`
```

Run the script and then check the contents of the directory. You should see that all pdfs have been changed to jpgs. The file `convertFiles-2.sh` contains the solution at this stage.

2 Demonstration 2: running a code and plotting results

Problem: You are commonly compiling and running code for a variety of different parameters to view the solution. You want to do this fast!

Solution: Write a script to create parameter input file, compile code, execute it, create pictures and open them up.

Download the zip file found here and extract into your home directory. The shell script solution is contained in `runCode.sh`. Try it out.

```
 #!/bin/bash
# compile and run a cpp file
# use gnuplot to plot the results

# create the parameter input file
# here " n m a b " are the values
echo " 25 25 3.4 1.5 " > parameters.in
```

```
# now compile the code
g++ myProblem.cpp -o myProb.out

# run the code, output from the code is "test.dat"
./myProb.out

# now plot the results, takes "test.dat" and create a 3D plot
# eps file myPlot.eps
gnuplot plotResults.plt

# take the eps and convert to pdf
epstopdf myPlot.eps -o myPlot.pdf

# open the results with a document viewer
evince myPlot.pdf
```

3 Demonstration 3: external access - running jobs

WARNING you must have maths username and password.

To access the maths from AT G.105 simply enter

```
ssh -X username@vummath.ma.man.ac.uk
```

in a terminal. You are now in the school servers! Exit from vummath by typing

```
exit
```

Back on your EPS machine try copying some files across to the server using the remote copy command, you could send over your bash configuration file

```
scp ~/.bashrc username@vummath.ma.man.ac.uk:
```

and the CodeRun directory

```
scp -r ~/CodeRun username@vummath.ma.man.ac.uk:
```

Now log back into vummath.

The default shell on vummath is tcsh so change to bash by simply typing

```
bash
```

and the terminal should appear as before. Now enter the CodeRun directory

```
cd CodeRun
```

and compile and run the code again:

```
g++ myProblem.cpp -o myProb.out
./myProb.out
```

If the program were to run for a long time you could run it in the background with the command

```
nohup ./myProb.out &
```

The proceeding command nohup makes sure that any output from the program is not sent to the screen causing the program to stop if you close the terminal.

You can also run scripts in the backup, so you could run the whole process

```
nohup ./runCode.sh &
```

in the background. There are other things you can do at the command line, for instance if you swap the evince command in `runCode.sh` for

```
# email the code when it has finished
# first create a message to send, and store in message.txt
echo "Hi, your program has finished, here is the result." > message.txt
# now use mutt command to send the email
# syntax is
# mutt [options] [email address] < [message file]
mutt -s "Code finished" -a myPlot.pdf \
                         paul.johnson-2@manchester.ac.uk < message.txt
```

The program mutt is a terminal email client, that allows you read and send emails from the command line. Using the command in this way will email you the picture when the code is finished. I have set up my university email account to allow IMAP and used a special configuration file `.muttrc` that allows me to send the email without entering a password. If you would like to know more about this just ask.

This could be very useful if you have a very long program running and you are out of the office!