CS214 Advanced UNIX

Assignment 2

Due: Friday, March 11, 11:59pm

At the end of this assignment you will

- gain experience writing more complex shell scripts, involving different types of loops
- understand how to use functions for breaking your code into modules
- understand how to apply certain operations to all files in a given directory, including the ones in subdirectories.

Personalized *man* command

If you look at the last example in the lecture notes 4, you will see that the script starts with some comments, placed in between two lines of the form

```
#Man entry
```

and

```
#End man entry
```

The comments in between these lines should give me enough information so that I recall what the script does, and how to invoke it. I would like now to create a new version of the **man** command that, when invoked with argument one of my scripts, displays the information I have written, and when invoked with argument one of the built-in commands in UNIX, returns the same information as the original **man** command. Write a script **myman.sh** that achieves this functionality.

Backup script

In many occasions you will want to make backup copies of all the files in a directory. One way to achieve this is by making sure that, at a given date, the system is making a copy of each file **foo** in your working directory. For simplicity, assume that the copy is named **foo.bak**. Write a script called **mybackup.sh** that, when given as argument a directory **dir**, creates backup copies of all files in **dir**; notice that we want the script to create backups for all files, in all subdirectories of **dir**, or subdirectories of subdirectories of **dir** and so on.

First assume that all the backups are stored in the same directory as the original files. Here is an example: assume that **dir** has one file **f1.txt**, and two subdirectories **dir1** and **dir2**, **dir1** has a file **f2.ps** and a subdirectory **dir1_1**, which in turn contains a file **f3.dvi**, and finally **dir2** has two files **f4.jpg** and **f5.txt**. By calling **mybackup.sh dir**, **dir** will have the files **f1.txt**, **f1.txt.bak**, subdirectories **dir1** and **dir2**, **dir1** will contain **f2.ps**, **f2.ps.bak** and subdirectory **dir1_1** with **f3.dvi** and **f3.dvi.bak**, and finally **dir2** will have **f4.jpg**, **f4.jpg.bak**, **f5.txt** and **f5.txt.bak**.

If we run twice **mybackup.sh dir**, we do not want to see backups of backups. In our example above, we do not want to see **f1.txt.bak.bak** in **dir**. This means that you have to test for each file whether a backup copy already exists. For example, suppose that I have done a backup copy of **f1.txt**, and then modified **f1.txt**. If I run **mybackup.sh dir**, I want **f1.txt.bak** to be a copy of the new **f1.txt**.

Next, allow the user to recover files from backups. To do so, the user will have to run the script with the option **-r** (from recovery). The user will then be prompted for each file whether he wants to recover the backup version or not. For example, the user will be asked

```
>f1.txt Recover? (y/n)
```

If the answer is "y", then copy **f1.txt.bak** back into **f1.txt**; if the answer is "n", then take no action. As the last step, allow the user the option to remove all the backups, say when trying to gain some space. To do so, the user has to run the script with the option **-c** and then all backup files are deleted. Make sure that if a file **foo.bak** exists, but no file **foo** exists, you do not delete **foo.bak**.

Test your script with different directories **dir**, and with a sequence of commands of the form

```
>mybackup.sh dir
>mybackup.sh dir
>mybackup.sh -r dir
>mybackup.sh dir
>mybackup.sh -c dir
```

Optional: What if now you would like to have a backup version of the whole hierarchy of directories and files. This would amount to constructing a new directory, called **dir.bak**, that would contain a file **f1.txt.bak**, subdirectories **dir1.bak** and **dir2.bak**, with **dir1.bak** containing **f2.ps.bak** and **dir1_1.bak** (with **f3.dvi.bak**), while **dir2.bak** would contain **f4.jpg.bak** and **f5.txt.bak**. Write a script **mybackup2.sh** with this functionality.

Recursive *grep*

You have already noticed that by invoking the command

```
>ls | grep '\.sh$'
```

you only search which files are ending in **.sh** in your current directory, without searching for such files in all subdirectories of your current directory and so on. Write a function **rgrep** that takes as argument a quoted expression (i.e. any quoted expression that can be passed as argument to **grep**) and searches all the files, recursively desceding in all subdirectories of the current directory. For example, if we assume that my current directory is **mycs214**, and it contains files **myman.sh** and **mybackup.sh**, plus a subdirectory **hw1** with files **test**, **ws.sh**, and **foo.ksh**, then a script in which I call

```
rgrep '\.sh$'
```

returns as output

```
myman.sh mybackup.sh ws.sh
```

Write a script called **recgrep.sh** in which you define the recursive function **rgrep** and you invoke it 3 times (you have the freedom to choose any 3 arguments that make sense as testing cases for your current directory).

What to turn in

Write your answers in 3 scripts: **myman.sh**, **mybackup.sh**, and **recgrep.sh**. Optionally, turn in **mybackup2.sh**.

The deadline is strict, unless you talk to me in advance and explain me the reason for your late submission.

How to get help

If you encounter problems, this is how you can get help:

- check the lecture notes for similar problems and their solutions
- check the newsgroup to see if other people have similar problems, or post your question
- come to office hours, and if you cannot make it, send me email for an appointment.

Good luck!