*Alternative OS – Lecture 5 – 19.10.2005*

Plan:

1. Recall redirecting standard input and output; file comparison
2. Comparing files
3. Wildcards
4. Conclusions

## 1. Redirecting standard input and output

Unix is a *file-based* OS. When a program is started, it opens three files:

- *Standard input* ***(stdin)***
- *Standard output* ***(stdout)***
- *Standard error* ***(stderr)***

One of the most useful features of the Unix OS is the ability to make commands read the *input* from somewhere else than the *keyboard* or write the *output* somewhere else than the *screen*. For example you can easily obtain a list of your working directory:

% **ls** > dir.txt

In this example you replaced the **stdout** file with the file **dir.txt.** Now the file *dir.txt* can be viewed using **less**:

% **less** *dir.txt*

or even changed using **vi**.

% **vi** *dir.txt*

Unix gives us a comfort of replacing us typing an input data for a program by a file of the data. Thus we can save time and energy if we have to input the same data many times.

If we happen to need the data sorted we can achieve it using **sort** command.

**For example**, imagine we want to have a list of the fifty richest gangsters of the town sorted by their names.

*born4prison.txt*

Donkey Jackass 0.5bn $
Goofy Bouncer 3.5bn $

Scrooge McDuck 1.5bn $
…

Then, we might change our mind and want them sorted by their fortunes. We can feed our file to the **sort** command:

% **sort +3 <** *born4prison.txt*

This will print the sorted list on the screen.

If we'd like to keep the output, we can redirect it to a file:

% **sort +3 <** *born4prison.txt* **>** *buxlist.txt*

In this example the *standard input* was replaced by the file *born4prison.txt* and the standard output – with the file *buxlist.txt*.

In order to *append* new lines to the existing file use **>>** for output redirection. For example:

% **sort +3 <** *born4prison.txt* **>>** *buxlist.txt*
% **cat** *buxlist.txt*

The **sort** command has got several useful options:

- **-f** - *ignore* difference between upper and lower case;
- **-n** - sort numbers *by their arithmetic values*, otherwise numbers are sorted according to their first digit;
- **-u** - *remove duplicate* lines;
- **-r** - sort in *reverse* order;
- **-t***x* - use *x* as the field separator instead of blanks.

## 2. Comparing files

We can compare files, if the alphabetical and the money-first lists are identical. There are several commands.

- **cmp** – to tell if the two files are different;
- **diff** - to see the ***diff***erences;
- **comm** – to see what's ***comm***on and what is different.

Usage examples:

% **cmp** *dir.txt sortdir.txt*

% **diff** *dir.txt sortdir.txt*

% **diff** *dir.txt sortdir.txt > diffdir.txt*

This command has several useful options:

1. **-b** - ignore changes in amount of *white space*;
2. **-B** - ignore *blank lines;*
3. **-q** - be *brief*, only report if the files differ or not;
4. **-i** - ignore changes in case;
5. **-y** - use *side by side* output format;
6. **-c** - use *context* output format.

If a *context* output format was called for, the **diff** command output may contain following characters at the left most column**:**

7. **!** – changed line
8. **-** - deleted line
9. **+** - added line

% **diff -*c*** *dir.txt sortdir.txt*

Another way of comparing file is to look for similarities rather than for differences. It shows common lines in one column and differing lines in separate columns for left and right files.

% **comm** *song.txt song2.txt* **>>** *dir.txt*

### 3. Power of computers

We might have noticed that computers automate routine tasks. Like those of sorting lists. In fact this is the most important improvement that they offer over the unarmed brains of humankind – the automation.

"Computers ain't intelligent, they just think they are."

But they perform so well where a routine task must be done. What other routine tasks do you know? Has it ever happened to you that you typed in lower case something you wanted in caps? Or maybe, you once forgot to switch on Cyrillic when typed something? Unless you had a nice tool to fix the mishap, you had to retype it all again. Have you ever tried to find a file in your hard drive you new you had but could never figure out where it could be? All these tasks can be done well by computers, and leave us doing somewhat more creative work!

## 4. Wildcards

Usually we *copy*, *move* or *delete* single files when we work with the *shell*. But there's more about the shell than this. Using *wildcards* we can manipulate *multiple* files simultaneously!

**Definition**

Strictly speaking, *wildcards* are special symbols that allow you to specify matches to *letters* or *letter sequences* in file names. Thus one *string* can be used to match more than one *filenames*.

- ***** matches zero or more characters of any kind

  **Examples:**

  % **ls** *.txt
  lists all files ending with ".txt" *extension,* that is *text files*.

  % **ls** a*
  lists all files that start with "a".

  % **rm** *
  removes *all* files in the current working directory.

  % **cp** public/* private/
  will copy all files from the *public*/directory to the *private*/ directory.

  % **cat** *poem.txt
  concatenates and prints all files ending with "poem.txt" to the display.

- **?** matches exactly one character of any kind.

  **Examples:**

% **cat** *ph?losophy.txt*

Concatenates and shows on the display files:

*philosphy.txt*
*phylosophy.txt*
*phklosophy.txt*
*phIlosophy.txt*
etc.

**Remark:**

This is useful when you don't remember how you called your file exactly!

% **ls** *report??.doc*

lists all files that start with "*report"* and end with two more symbols plus ".doc" *extension:*

*report01.doc*
*report02.doc*
…
*report12.doc*
*reportaa.doc*

But it would not list files:

*report1.doc*
*reporta.doc*
*report.doc*

% **rm** ***?***

removes all one-character named files:

*1*
*x*
*7*
etc.

- **[ ]** matches one of the characters inside the brackets

**Examples:**

% **more** *[Rr][Ee][Aa][Dd][Mm][Ee]*

Opens all *Readme, README, ReadMe, ReAdMe* etc. files in the **more** viewer.

% **ls** *.[a-z]

Lists all files with one-letter *extension:*

*contract.a*

*abstract.b*
*123.t*
*story.h*
*hello.c*

**Remark:** it would not list files:
*contract.1*
*123.4*
*hello.pas*

When using *wildcards* with shell commands that do not work with multiple arguments, like **cd** that can choose only one directory, make sure that your *wildcard* filename will not be ambiguous.

**For example:**

You know you had a directory called either *public* or *publik*. You don't remember how you called it exactly but you are sure there's only one directory *matching* "*publi?"* then

% **cd** *publi?*

works just fine. However, if you issue a command like

% **cd /***

you'll get an error message: "**/*: Ambiguous.**"

## 5. Conclusions

Computers allow us to perform routine tasks easily. Since they are sort of imagination, this is almost as far as they can fare. In Unix two kinds of routine tasks can be easily automated:

- File comparison using **cmp**, **diff**, and **comm**,
- File manipulation, using *wildcards*.