

Modelling, Uncertainty and Data for Engineers (MUDE)

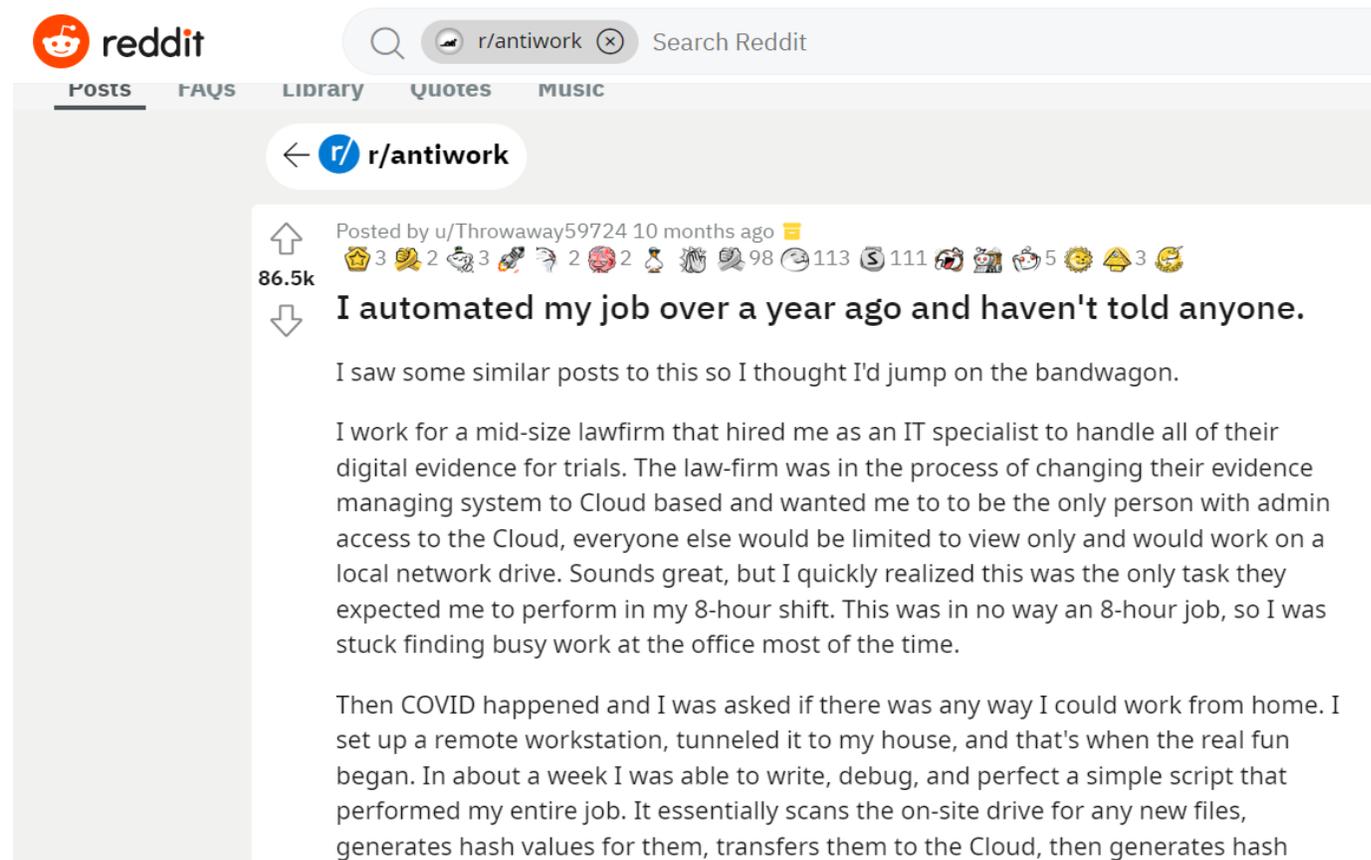
Basic Linux shell commands

Denis Voskov, Ilshat Saifullin

Shell introduction

- Shell is ...
 - command-line interface
 - scripting language
- Most popular is Bash
- No clicks as in GUI. Just write the commands and execute them
- Perfect for repetitive tasks
- Combine scripts and organize your workflow

Example of automation



reddit

Search Reddit

POSTS FAQS LIBRARY QUOTES MUSIC

← r/antiwork

Posted by u/Throwaway59724 10 months ago

86.5k

I automated my job over a year ago and haven't told anyone.

I saw some similar posts to this so I thought I'd jump on the bandwagon.

I work for a mid-size lawfirm that hired me as an IT specialist to handle all of their digital evidence for trials. The law-firm was in the process of changing their evidence managing system to Cloud based and wanted me to to be the only person with admin access to the Cloud, everyone else would be limited to view only and would work on a local network drive. Sounds great, but I quickly realized this was the only task they expected me to perform in my 8-hour shift. This was in no way an 8-hour job, so I was stuck finding busy work at the office most of the time.

Then COVID happened and I was asked if there was any way I could work from home. I set up a remote workstation, tunneled it to my house, and that's when the real fun began. In about a week I was able to write, debug, and perfect a simple script that performed my entire job. It essentially scans the on-site drive for any new files, generates hash values for them, transfers them to the Cloud, then generates hash

https://www.reddit.com/r/antiwork/comments/s2igq9/i_automated_my_job_over_a_year_ago_and_havent/?sort=top

Shell command syntax

- General syntax

COMMAND OPTIONS ARGUMENTS

- Getting help

- COMMAND --help
- man COMMAND
- online help <http://man.he.net>

Example

- ls
- ls -- help
- ls -l *.txt

```
lcur1922@r11n24:~$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
-a, --all                do not ignore entries starting with .
-A, --almost-all       do not list implied . and ..
                        --author          with -l, print the author of each file
-b, --escape             print C-style escapes for nongraphic characters
                        --block-size=SIZE with -l, scale sizes by SIZE when printing them;
                        e.g., '--block-size=M'; see SIZE format below
-B, --ignore-backups    do not list implied entries ending with ~
-c                       with -lt: sort by, and show, ctime (time of last
                        modification of file status information);
                        with -l: show ctime and sort by name;
                        otherwise: sort by ctime, newest first
-C                       list entries by columns
--color[=WHEN]          colorize the output; WHEN can be 'always' (default
                        if omitted), 'auto', or 'never'; more info below
-d, --directory         list directories themselves, not their contents
-D, --dired              generate output designed for Emacs' dired mode
-f                       do not sort, enable -aU, disable -ls --color
-F, --classify          append indicator (one of */=>@|) to entries
                        likewise, except do not append '*'
--file-type             across -x, commas -m, horizontal -x, long -l,
--format=WORD           single-column -l, verbose -l, vertical -C
--full-time             like -l --time-style=full-iso
-g                       like -l, but do not list owner
--group-directories-first
```

Shell keys

Key	Description
Up	Previous command
Ctrl+R then type	Search in command history
Ctrl+C	Interrupt the command
Tab	Autocomplete the command or filename

Shell commands: files and directories

Commandls	Description	Examples		
<code>ls</code>	List current directory	<code>ls</code>	<code>ls ./logs/1?.txt</code>	<code>ls -l</code>
<code>pwd</code>	Print current directory path	<code>pwd</code>		
<code>cd DIR</code>	Change directory	<code>cd ~</code>	<code>cd ..</code>	<code>cd /usr/include</code>
<code>mkdir DIR</code>	Make a new directory	<code>mkdir data</code>		
<code>cp SRC DEST</code>	Copy a file or directory	<code>cp 1.txt 2.txt</code>	<code>cp *.txt ./logs</code>	<code>cp -r dir_1 dir_2</code>
<code>mv SRC DEST</code>	Move/rename a file or directory	<code>mv old.txt new.txt</code>	<code>mv new.txt data</code>	
<code>rm FILE</code>	Remove the file or directory. Note: removed files cannot be	<code>rm new.txt</code> <code>rm log*.txt</code> <code>rm -r data</code>		

Shell commands: files and directories

Command	Description	Examples
touch <i>FILE</i>	Create an empty file with name <i>FILE</i>	touch my_file.txt
which <i>FILE</i>	Print the path to executable	which python
tar -czf <i>ARCH INPUT</i>	Create an archive with name <i>ARCH</i>	tar czf data.tar.gz data data_2
tar -xzf <i>ARCH</i>	Unpack the archive	tar xzf data.tar.gz
chmod <i>MODE FILE</i>	Change mode for the <i>FILE</i> MODE: u/g/o/a +/- r/w/x	chmod a+x *.sh
find	Finds files by name pattern, type, ...	find mydir/'pattern'

Shell commands: tools

Command	Description	Examples
<code>echo <i>TEXT</i></code>	Display a line of text	<code>echo "Hello!"</code>
<code>cat <i>FILE_1 FILE_2 ...</i></code>	Concatenate files and print to std output	<code>cat cities.txt</code> <code>cat 1.txt 2.txt</code>
<code>head -n 2 <i>FILE</i></code>	Output the first # lines of file(s)	<code>head -n 10 cities.txt</code>
<code>tail -n 2 <i>FILE</i></code>	Output the last # lines of file(s)	<code>tail -n 10 *.txt</code>
<code>wc <i>FILE</i></code>	Print line, word, byte counts for file(s)	<code>wc -l cities.txt</code>
<code>grep <i>TEXT FILE</i></code>	Print lines of file that match patterns	<code>grep Den Haag cities.txt</code>

Shell commands: pipes

Command	Description	Examples
COMMAND > FILE	Redirect output to the FILE	cat 1.txt 2.txt > 12.txt
COMMAND >> FILE	Redirect output to the FILE in append mode	cat 1.txt >> 12.txt cat 2.txt >> 12.txt
COMMAND < FILE	Use FILE as input to COMMAND	tail -n 10 *.txt
CMD1 CMD2	Use output of the command CMD1 as input to the command CMD2	cat cities.txt grep Rotterdam

Shell: Variables

- Shell variables are treated as strings
- Variables are assigned using "=" symbol
- Add "\$" before the name of the variable to use it
- The PATH variable defines the shell's search path

Example:

```
a=1
b=5
c=a+b
c=$((a + b))
```

Command	Description
<code>echo \$PATH</code>	Show the PATH variable value
<code>s="Hello"</code> <code>echo \$s</code>	Set a variable and print the value.
<code>export s="Hello"</code>	Set as an environment variable . Other processes started from this shell will use it.

Shell: loops

```
for f in *.txt
do
    echo $f
done
```

```
for f in *.txt; do echo $f; done
```

```
for f in *.txt
do
    tail -n 10 $f >> all.txt
done
```

```
for entity in temp pressure
do
    for value in 5 10 15
    do
        echo $entity $value
    done
done
```

```
for i in $(seq 5 10); do echo $i; done
```

Shell: conditions

```
if CONDITION
then
COMMANDS_1
else
COMMANDS_2
fi
```

```
# spaces in [ ] are mandatory
if [ $i == 5 ]
then
echo "I is 5"
else
echo "I is not 5"
fi
```

Operator	Description
&&	logical and
	logical or
-gt	> (greater)
-ge	>= (greater or equal)

```
# spaces in [ ] are mandatory
if [ $i -gt 5 ]
then
echo "I greater than 5"
fi
```

Shell: system information

Command	Description
<code>cat /proc/cpuinfo</code>	Processor info
<code>uname -a</code>	OS info
<code>lsb_release -a</code>	OS version
<code>df -h</code>	Free space on disk
<code>free</code>	Amount of free and used memory (RAM)
<code>top -c</code>	Processes (-c to show command arguments)

Shell: processes and signals

Command	Description
<code>./my_script.sh</code>	use <code>./</code> to run executable file from current directory (if not in <code>\$PATH</code>)
<code>time ./my_script.sh</code>	Run and measure the resource usage

```
touch my_script.sh
chmod a+x my_script.sh
./my_script.sh
time ./my_script.sh
```

Text Editor (nano)

Command	Description
<code>nano my.txt</code>	Open the editor
<code>Ctrl+G</code>	Show Help
<code>Ctrl+O</code>	Save the file
<code>Ctrl+X</code>	Exit

```
GNU nano 6.2                               New Buffer

^G Help      ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File  ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Bash script file

1. `#!/bin/bash` at the first line
(interpreter)

```
GNU nano 6.2
#!/bin/bash
echo "Hello"
```

2. Using cmd arguments: `$#`, `$1`

```
echo "Script $0 is running with $# arguments."
echo "The first argument is $1"
```

3. Commenting: `#`

```
user@TUD258093:~$ ./my_script.sh 50
Script ./my_script.sh is running with 1 arguments.
The first argument is 50
```

Exercise 1: count files and run script

1. Create a text file with your bash commands
2. Depending on the argument, print the number of files in directory
 - a) If no arguments: in directory `Week_2_2` (recursively)
 - b) Argument is `sub_dir`: in directory `Week_2_2 /sub_dir`
3. Check that the sum of results of two calls with arguments `well_logs` and `well_tests` is equal to result of call without arguments. Use variables.

```
user@TUD258093:~$ ./files_count.sh
Script ./files_count.sh is running with 0 arguments.
55
user@TUD258093:~$ ./files_count.sh well_logs
Script ./files_count.sh is running with 1 arguments.
The first argument is well_logs
30
user@TUD258093:~$ ./files_count.sh well_tests
Script ./files_count.sh is running with 1 arguments.
The first argument is well_tests
25
```

Exercise 2: search, print and collect the data

1. Depending on the argument **well_name**, print the temperature from file `Week_2_2 /well_tests/well_name_welltest.txt`
2. Collect all the filenames and well temperatures to one file

```
/mnt/c/Users/isaifullin/soft~/well_tests/W10_welltest.txt
@@@ Well-test report, powered by DARTS (https://darts.citg.
@@@          Special edition for MUDE 2022-2023
Well name: W10
Perforation interval: 1998 to 2074
Fluid in place: brine
Initial pressure:      201.035
Temperature:          344.440
```

```
Temperature:          339.194
4shell_excercise/well_tests/W20_welltest.txt
Temperature:          346.606
4shell_excercise/well_tests/W21_welltest.txt
Temperature:          349.855
4shell_excercise/well_tests/W22_welltest.txt
Temperature:          338.977
4shell_excercise/well_tests/W23_welltest.txt
Temperature:          352.395
4shell_excercise/well_tests/W24_welltest.txt
Temperature:          344.258
```

Exercise 3: use archive and process files

1. Create an archive with `well_test` files for the wells with names: `W3`, `W5`, `W15`, `W17`.
Files are in the directory:
`well_tests/well_name_welltest.txt`
2. Create a new directory ***my_data*** and copy all the files from `Week_2_2` directory to it. Remove the files for the wells `W1`, `W2`, .. , `W10` in ***my_data***