Scripting Languages: Workshop 2

Pre-requisite:

If you have not already done so, log into your Azure Linux VM, start VS Code and navigate your way into the **`ws2`** folder.

Task 1

Examine the table below **carefully**. It contains a list of the bash commands, utilities and keywords (in alphabetical order) that you **will** draw upon when constructing your shell scripts for workshop tasks and your assessments.

If you visit and bookmark https://ss64.com/bash/, you will find concise descriptions for each of the commands in the table (perhaps easier than using the **`man`** pages)

bash Commands, Keywords and Utilities

Name	Type	Function
awk	utility	Non-compiled scripting language utility used for simple to complex data manipulation data and report generation
basename	command	Strips directory and suffix from filenames
bc	utility	Arbitrary precision calculator language
break	keyword	Exit from a for, while, or until loop early
case	keyword	Conditionally perform a command
cat	command	Concatenate and print (display) the content of files
chmod	command	Change access permissions to files and folders
clear	command	Clear terminal screen of all text
continue	keyword	Skip to the next iteration of a running for, while, or until loop
cp	command	Copy one or more files/folders to another location
cut	command	Rules-based division of a file's or variable's argument resident data
curl	utility	Transfer data from or to a server
date	command	Extract, display or change the date/time values
declare	keyword	Declare variables or arrays with pre-defined data types/attributes
du	command	Estimate file space usage of a file or system volume
echo	command	Display textual output to the terminal
exit	command	Exit the shell
expr	command	Evaluate stated expressions and returns a result
false	keyword	Indicate false result to test, or to do nothing
find	command	Search for files that meet a desired criteria
for	keyword	Indicates to execute commands on items in file or array one-by-one
function	keyword	Define a function
getopts	command	Parses command options and arguments, often those passed to a shell script from the command line
grep	utility	Search file(s) for lines that match a given pattern
if	keyword	Conditionally perform a command
less	command	Display output one screen at a time
let	keyword	Perform arithmetic on shell variables

Name	Type	Function
local	keyword	Create a function variable
ls	command	List information about file/folders
man	command	Help manual
mkdir	command	Create/make a new directory (folder)
more	command	Display output one screen at a time
mv	command	Move or rename files or directories
printf	command	Format and print data to output, e.g. terminal, file, variable
pwd	command	Prints path to Working Directory to terminal or other output
read	keyword	Read a line from standard input
return	keyword	Exit a shell function, usually with a return value
rev	command	Reverse lines of a file
rm	command	Remove files/folders
rmdir	command	Remove folder(s) (only if empty)
sed	utility	Stream Editor, a.ka. search and replace
seq	command	generate numbers from FIRST to LAST in steps of INCREMENT
shuf	command	Generate random permutations of provided arguments
sort	command	Sort text files according to stipulated criteria
tail	command	Output last specified lines a file, default is 10 lines
tee	command	Redirect output to multiple outputs, e.g., terminal and file
test	keyword	Evaluate a conditional expression
touch	command	Create a new file or update an existing file's timestamp
tr	command	*Translate*, squeeze, and/or delete characters
true	keyword	Indicate true result to test, or trigger to do something
uniq	command	Filters out duplicate items in a file
unset	keyword	Remove variable or function names
until	keyword	Indicates command execution until a criteria is no longer true
wc	command	Count the bytes, words, or lines in a file or variables contents
which	command	Search the user's $path for a program file
while	keyword	Indicates command execution until a criteria is no longer false
zip	command	File compression and packaging utility

Task 2

1. Write a shell script named **`intro.sh`** to which three (3) arguments will be passed when run at the command line
2. The three (3) arguments are to be 1) your *first name*, 2) your *age*, and 3) the *year* you anticipate graduating from ECU (see the image below):

```
[redacted]:~/scrlang/workshops/ws2/sol$ ./intro.sh Vince 53 2021
My name is Vince and I am 53 and I will graduate from ECU in 2021
```

3. Be sure to **not** use your own defined variable names; use the *default* shell script variables described in this week's lecture video instead
4. Run the script and make sure it prints the correct output to the terminal (don't forget to give the script execute permissions)

Task 3

1. Write a shell script named **`autofolder.sh`** to which two (2) arguments will be passed when run at the command line
2. The two (2) arguments are to be 1) the directory name **`user1`**, and 2) a text file named **`profile.txt`**
3. Ensure that the **`profile.txt`** file is placed into the **`user1`** directory
4. Be sure to **not** use your own defined variable names; use the *default* shell script variables instead
5. Run the script and make sure it prints the correct output to the terminal, using **`ls`** and **`ls user1`** to ensure that both the directory and file have been created, and that the file is in the directory (see image below – be sure to give **`autofolder.sh`** execute permissions)

```
[redacted]:~/scrlang/workshops/ws2/sol$ ./autofolder.sh user1 profile.txt
The [user1] directory has been created and populated with the file [profile.txt]
```

Task 4

1. **Copy** the *`.sh`* files you created in today's workshop to the *backups* directory using the same *_bu* name modification you used in last week's workshop
2. Navigate to the *backups* directory and make sure the copy procedure was successful

Conclude:

Close the *RDP connection* to your Azure Linux VM and then **power off** your VM in Azure.