January 2023 Cohort, welcome to

Shell I/O, Redirections and Filters

Just Before We Get Started

House-Keeping Rules

Let's Get Started

HouseKeeping Rules

- Stay attentive, we do not want YOU missing **anything.**
- Share your questions on the **Slido link**
- **Links and instructions** to participate in activities **will be shared in the Youtube chat**

Agenda

Activity	Duration(EAT)
Check in and recap	3:00 PM
Shell I/O, Redirection and Filters	3:10 PM
Q & A Session	3:40 PM
Announcements & Next steps	4:00 PM

Pay close attention, we are now starting our:

Shell I/O Redirection and Filters

I/O Redirections

- Many shell commands print their output on the display/terminal.
- In shell programming, there are different ways to provide an input (for example, via a keyboard and terminal) and display an output (for example, terminal and file) and error (for example, terminal), if any, during the execution of a command or program.

Standard input stream (stdin)

In Bash, the standard input stream (stdin) is a source of input data for commands and scripts.

It is typically connected to the keyboard, but can be redirected to a file or another command's output using the "<" or "|" operators. Commands can read input from stdin using the "read" command or by simply accepting input from the shell. By default, the stdin file descriptor is represented by the number 0.

> output redirector

<, | input redirector

Standard output stream (stdout)

the standard output stream (stdout) is the default destination for a command's output. By default, stdout is directed to the terminal window, but it can be redirected to a file or another command's input using the ">" or "|" operators. Commands can send output to stdout using the "echo" command, or by printing output to the console using the "printf" command. By default, the stdout file descriptor is represented by the number 1

Standard Error Stream (stderr)

- the standard error stream (stderr) is used to output error messages and diagnostics from commands and scripts. By default, stderr is directed to the terminal window, separate from stdout, but it can be redirected to a file or another command's input using the "2>" or "2|" operators. Commands can send error messages to stderr using the "echo" command with the "-e" or "-n" options, or by printing error messages to the console using the "printf" command. By default, the stderr file descriptor is represented by the number 2.

In Short

When a program executes, by default, three files get opened with it which are *stdin*, *stdout*, and *stderr*. The following table provides a short description about them:

File Descriptor Number	File Name	Description
0	stdin	This is standard input being read from the terminal
1	stdout	This is standard output to the terminal
2	stderr	This is standard error to the terminal

Redirecting the standard I/O and error streams

- We have an option to redirect standard input, output, and errors, for example, to a file, another command, intended stream, and so on

Operator	Description
>	This redirects a standard output to a file
>>	This appends a standard output to a file
<	This redirects a standard input from a file
>&	This redirects a standard output and error to a file
>>&	This appends a standard output and error to a file
|	This redirects an output to another command

Redirecting standard output

- An output of a program or command can be redirected to a file. Saving an output to a file can be useful when we have to look into the output in the future.

``$ ls -la > output.txt
``$ echo "I am redirecting output to a file" > output.txt

``$ cat output.txt
``$ I am redirecting output to a file

Note: Demonstrate `>>`

Filters

filters are commands that process and modify text input before passing it along to other commands or outputting it to the screen. Some common filters include:

- grep: searches for specific patterns in text input
- sed: performs search-and-replace operations on text input
- awk: processes and manipulates text input based on fields and patterns
- cut: extracts specific columns or fields from text input
- sort: sorts lines of text input alphabetically or numerically
- uniq: removes duplicate lines from text input
- tr: translates or deletes characters in text input

...among others

- The head and tail commands are file-oriented commands in Unix and Unix-like operating systems.
    - The head command is used to print the first few lines (by default, the first 10 lines) of a file or set of files. It is often used to preview the contents of a large file.
    - The tail command, on the other hand, is used to print the last few lines (by default, the last 10 lines) of a file or set of files. It is often used to monitor the output of a continuously updating file, such as a log file.
- Both head and tail are commonly used in command pipelines to extract a specific range of lines from a file or as part of more complex commands to process text or data.

Redirecting standard input

- Instead of getting an input from a standard input to a command, it can be redirected from a file using the < (less than) operator. For example, we want to count the number of words in the *output.txt* file

```
$ wc -w < output.txt
```

Redirecting standard error

- There is a possibility of getting an error while executing a command/program in bash because of different reasons such as invalid input, insufficient arguments, file not found, bug in program, and so on:

``$ cd /homE # Trying to cd into a directory that is not existing in that path
bash: cd: /victor: No such file or directory

=> Bash prints the error on a terminal saying the directory is non-existing

``$ cd /homE 2> error.txt

The following command redirects both *stdout* and *stder*:

```
bash-3.2$ (ls ~/Demo/ ; cat hello.txt;) > log.txt 2>&1
bash-3.2$ cat log.txt
error.txt
hi-there
iacta
items.txt
log.txt
output.txt
password.txt
sorted_items.txt
cat: hello.txt: No such file or directory
bash-3.2$ _
```

Pipe

- The pipe denoted by the operator | connects the standard output of a process in the left to the standard input in the right process by inter process communication mechanism. In other words, the | (pipe) connects commands by providing the output of a command as the input to another command.

Demo:
``$ cat /etc/passwd | less

Explanation: the *cat* command, instead of displaying the content of the */proc/cpuinfo* file on *stdout*, passes its output as an input to the *less* command. The *less* command takes the input from *cat* and displays on the *stdout* per page.

Pipeline

- Pipeline is a sequence of programs/commands separated by the operator ‘ | ‘ where the output of execution of each command is given as an input to the next command. Each command in a pipeline is executed in a new subshell. The syntax will be as follows:

Demo
``$ cd ~/Demo/ | cat items.txt | grep HDD

stdin
command <items.txt
stdout
ls > examplelist.txt

stderr

Announcements

Do not forget to join the next live learning session

See you at the next session!