How to Type Less with Bash

Because typing sucks

PA Hackers
June 25, 2019

Mike Salvatore
Ubuntu Security Engineer

CANONICAL ubuntu

Introduction

About Me

- Education
  - B.S. Electrical Engineering, Rutgers University
  - M.S. Cybersecurity, Johns Hopkins University
- Current employment
  - Ubuntu Security Team at Canonical
- Contact
  - mike.salvatore@canoncal.com
  - msalvatore on the freenode IRC network
  - PA Hackers Slack

Overview

Using the history

Bash key bindings and other shortcuts

Scripts, Aliases, and Functions

Some security implications

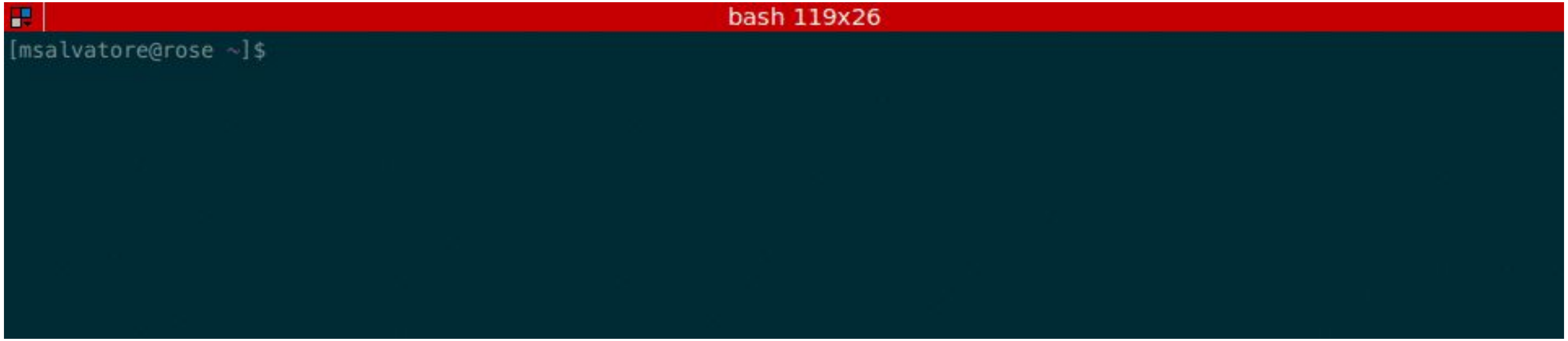
**Note: Most of the value of this presentation is in making you aware of these shortcuts. You have to practice them to integrate them into your workflow. I recommend you print out a cheat sheet and stick it to the wall where you most frequently work. I also recommend you read the links in this presentation. They contain a wealth of information.*

Using the History

Using the History: Up/Down Arrow, Ctrl+r

The up and down arrows on your keyboard will allow you to cycle through previous commands.

The Ctrl+r key sequence allows you to search through the command history.



```
bash 119x26
[msalvatore@rose ~]$
```

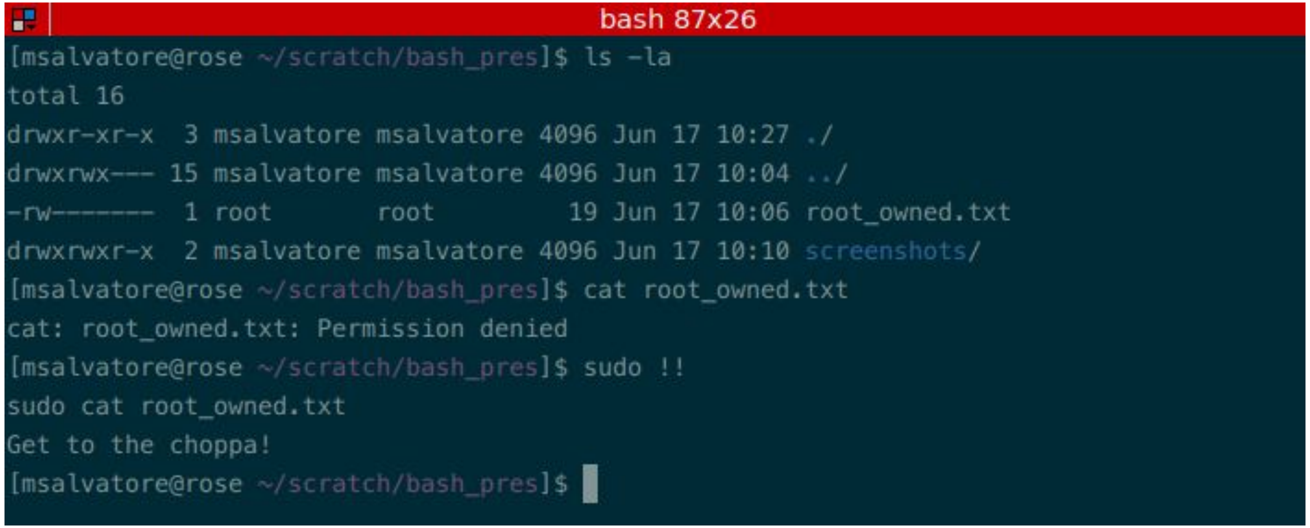
Using the History: Event Designators

An event designator is a reference to a command line entry in the history list.

This section contains some of the event designators that I use most often. For more event designators, see https://www.gnu.org/software/bash/manual/html_node/Event-Designators.html

Using the History: Event Designators — !!

The `!!` event designator refers to the previous command.

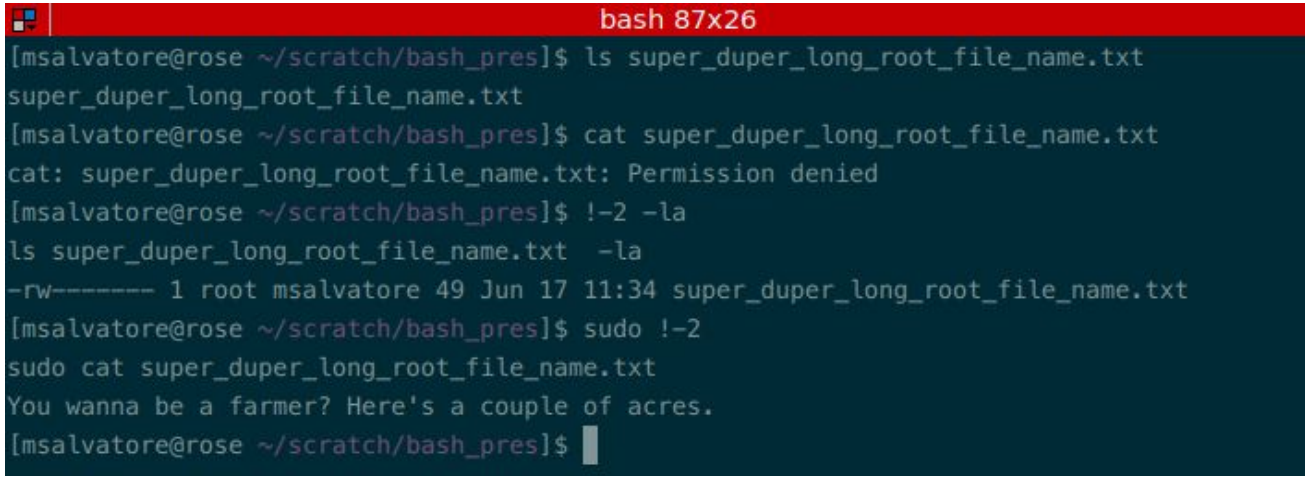


bash 87x26

```
[msalvatore@rose ~/scratch/bash_pres]$ ls -la
total 16
drwxr-xr-x  3 msalvatore msalvatore 4096 Jun 17 10:27 ./
drwxrwx--- 15 msalvatore msalvatore 4096 Jun 17 10:04 ../
-rw-------  1 root       root         19 Jun 17 10:06 root_owned.txt
drwxrwxr-x  2 msalvatore msalvatore 4096 Jun 17 10:10 screenshots/
[msalvatore@rose ~/scratch/bash_pres]$ cat root_owned.txt
cat: root_owned.txt: Permission denied
[msalvatore@rose ~/scratch/bash_pres]$ sudo !!
sudo cat root_owned.txt
Get to the choppa!
[msalvatore@rose ~/scratch/bash_pres]$
```

Using the History: Event Designators — !-n

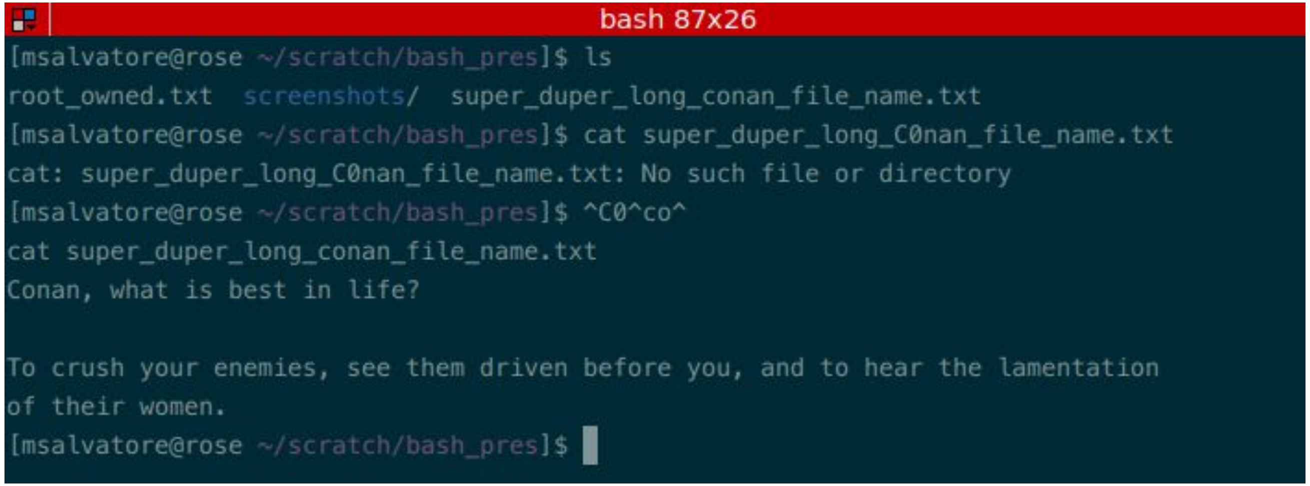
The `!-n` event designator refers to the command *n* lines back



```
[msalvatore@rose ~/scratch/bash_pres]$ ls super_duper_long_root_file_name.txt
super_duper_long_root_file_name.txt
[msalvatore@rose ~/scratch/bash_pres]$ cat super_duper_long_root_file_name.txt
cat: super_duper_long_root_file_name.txt: Permission denied
[msalvatore@rose ~/scratch/bash_pres]$ !-2 -la
ls super_duper_long_root_file_name.txt  -la
-rw------- 1 root msalvatore 49 Jun 17 11:34 super_duper_long_root_file_name.txt
[msalvatore@rose ~/scratch/bash_pres]$ sudo !-2
sudo cat super_duper_long_root_file_name.txt
You wanna be a farmer? Here's a couple of acres.
[msalvatore@rose ~/scratch/bash_pres]$ 
```

Using the History: Event Designators — Substitution

^string1^string2^ will repeat the last command, replacing the first occurrence of string1 with string2.

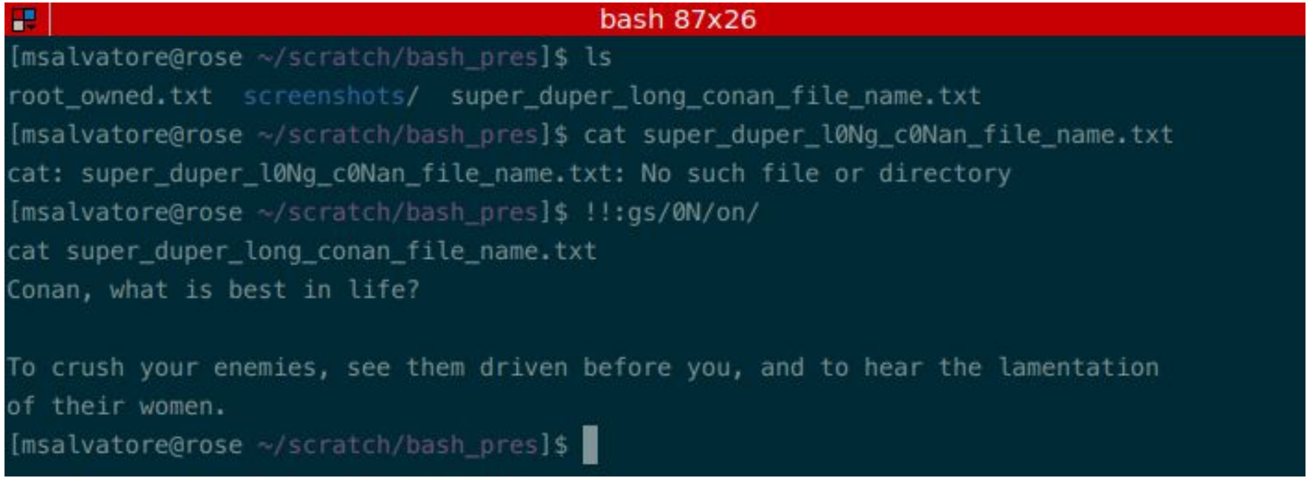


bash 87x26

```
[msalvatore@rose ~/scratch/bash_pres]$ ls
root_owned.txt  screenshots/  super_duper_long_conan_file_name.txt
[msalvatore@rose ~/scratch/bash_pres]$ cat super_duper_long_C0nan_file_name.txt
cat: super_duper_long_C0nan_file_name.txt: No such file or directory
[msalvatore@rose ~/scratch/bash_pres]$ ^C0^co^
cat super_duper_long_conan_file_name.txt
Conan, what is best in life?

To crush your enemies, see them driven before you, and to hear the lamentation
of their women.
[msalvatore@rose ~/scratch/bash_pres]$ 
```

Using the History: Event Designators — Substitution

!!:gs/string1/string2/ will repeat the last command, globally replacing string1 with string2.



bash 87x26

```
[msalvatore@rose ~/scratch/bash_pres]$ ls
root_owned.txt  screenshots/  super_duper_long_conan_file_name.txt
[msalvatore@rose ~/scratch/bash_pres]$ cat super_duper_l0Ng_c0Nan_file_name.txt
cat: super_duper_l0Ng_c0Nan_file_name.txt: No such file or directory
[msalvatore@rose ~/scratch/bash_pres]$ !!:gs/0N/on/
cat super_duper_long_conan_file_name.txt
Conan, what is best in life?

To crush your enemies, see them driven before you, and to hear the lamentation
of their women.
[msalvatore@rose ~/scratch/bash_pres]$ 
```

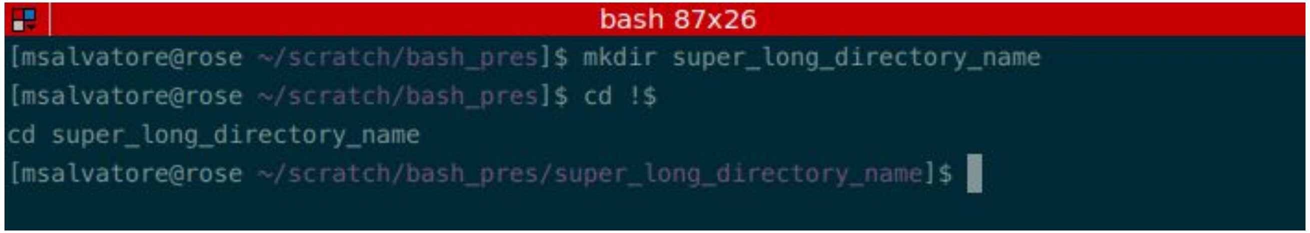
Using the History: Word Designators

"Word designators are used to select desired words from the event. A ':' separates the event specification from the word designator. It may be omitted if the word designator begins with a '^', '$','*', '-', or '%'. Words are numbered from the beginning of the line, with the first word being denoted by 0 (zero). Words are inserted into the current line separated by single spaces."

This section contains some of the word designators that I use most often. For more word designators, see https://www.gnu.org/software/bash/manual/html_node/Word-Designators.html

Using the History: Word Designators - !$

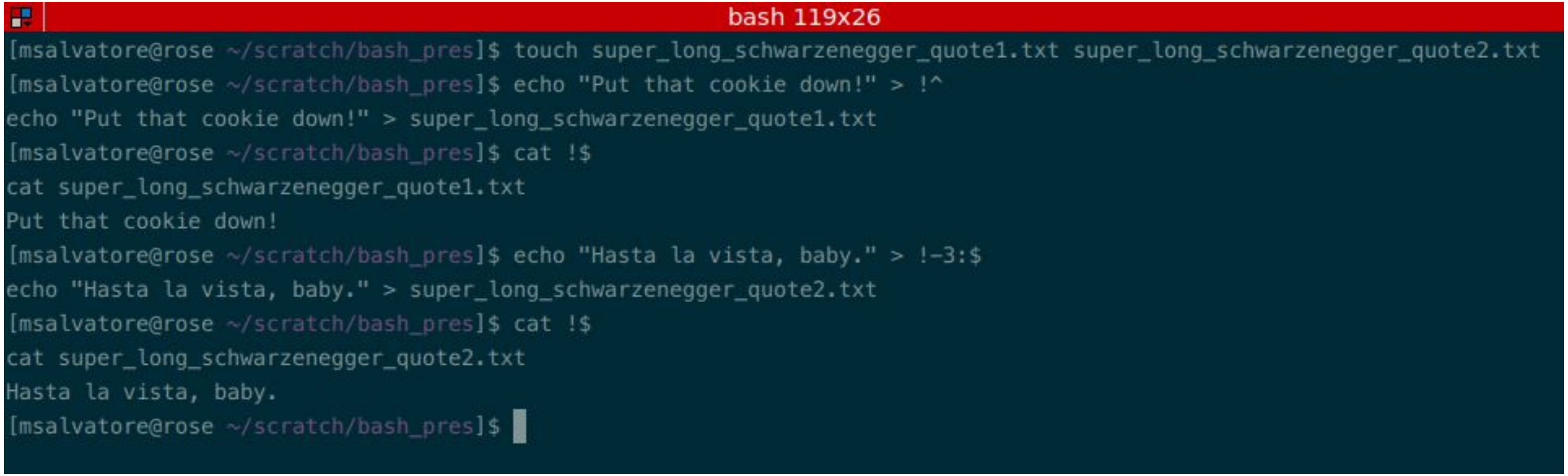
`!!:$` or `!$` refer to the last argument of the preceding command



```
[msalvatore@rose ~/scratch/bash_pres]$ mkdir super_long_directory_name
[msalvatore@rose ~/scratch/bash_pres]$ cd !$
cd super_long_directory_name
[msalvatore@rose ~/scratch/bash_pres/super_long_directory_name]$
```

Using the History: Word Designators - !^

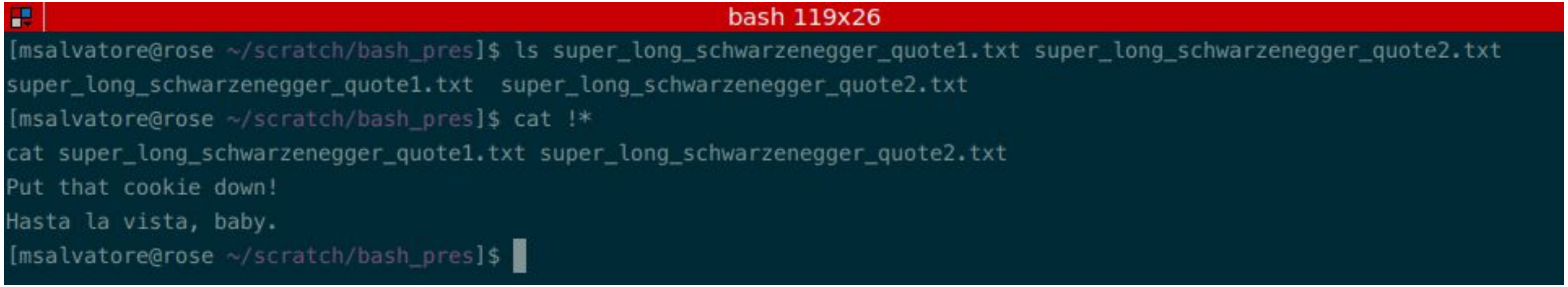
`!^` refers to the first argument of the preceding command (word 1)



```
bash 119x26
[msalvatore@rose ~/scratch/bash_pres]$ touch super_long_schwarzenegger_quote1.txt super_long_schwarzenegger_quote2.txt
[msalvatore@rose ~/scratch/bash_pres]$ echo "Put that cookie down!" > !^
echo "Put that cookie down!" > super_long_schwarzenegger_quote1.txt
[msalvatore@rose ~/scratch/bash_pres]$ cat !$
cat super_long_schwarzenegger_quote1.txt
Put that cookie down!
[msalvatore@rose ~/scratch/bash_pres]$ echo "Hasta la vista, baby." > !-3:$
echo "Hasta la vista, baby." > super_long_schwarzenegger_quote2.txt
[msalvatore@rose ~/scratch/bash_pres]$ cat !$
cat super_long_schwarzenegger_quote2.txt
Hasta la vista, baby.
[msalvatore@rose ~/scratch/bash_pres]$
```

Using the History: Word Designators - !*

`!*` refers to all argument of the preceding command, except argument 0



bash 119x26

```
[msalvatore@rose ~/scratch/bash_pres]$ ls super_long_schwarzenegger_quote1.txt super_long_schwarzenegger_quote2.txt
super_long_schwarzenegger_quote1.txt  super_long_schwarzenegger_quote2.txt
[msalvatore@rose ~/scratch/bash_pres]$ cat !*
cat super_long_schwarzenegger_quote1.txt super_long_schwarzenegger_quote2.txt
Put that cookie down!
Hasta la vista, baby.
[msalvatore@rose ~/scratch/bash_pres]$ 
```

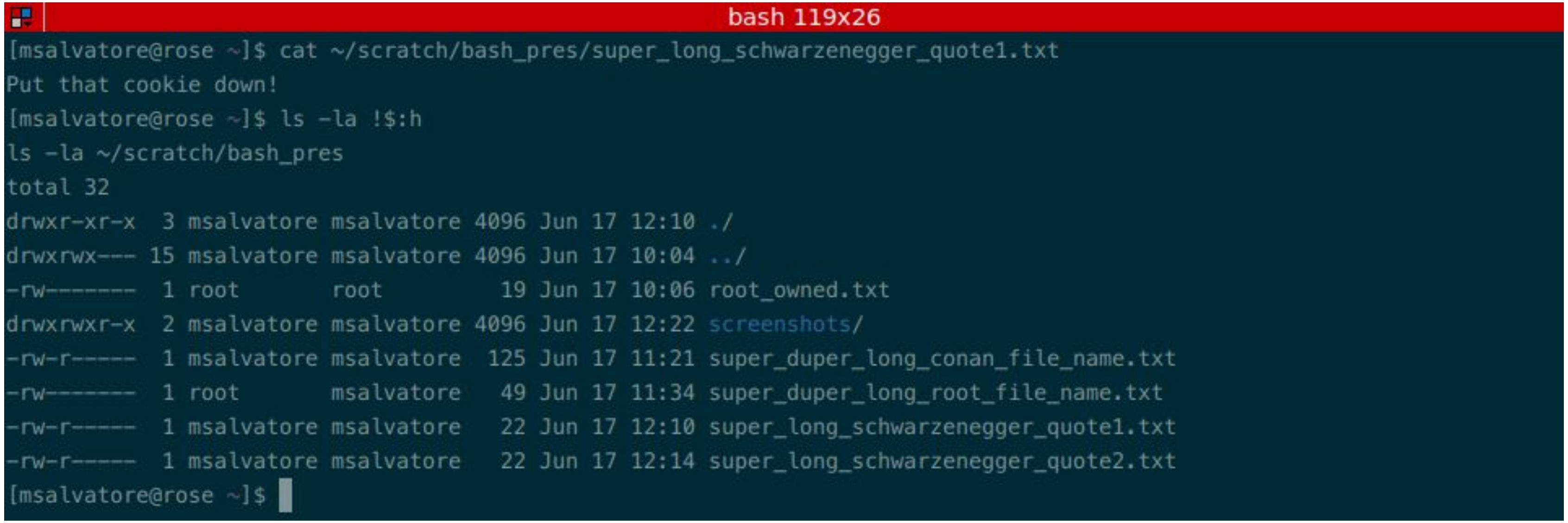
Using the History: Modifiers

Modifiers change word designators. Each modifier is preceded by a ':'.

This section contains some of the word designators that I use most often. For more modifiers, see
https://www.gnu.org/software/bash/manual/html_node/Modifiers.html#Modifiers

Using the History: Modifiers — h

Remove a trailing pathname component, leaving only the head.

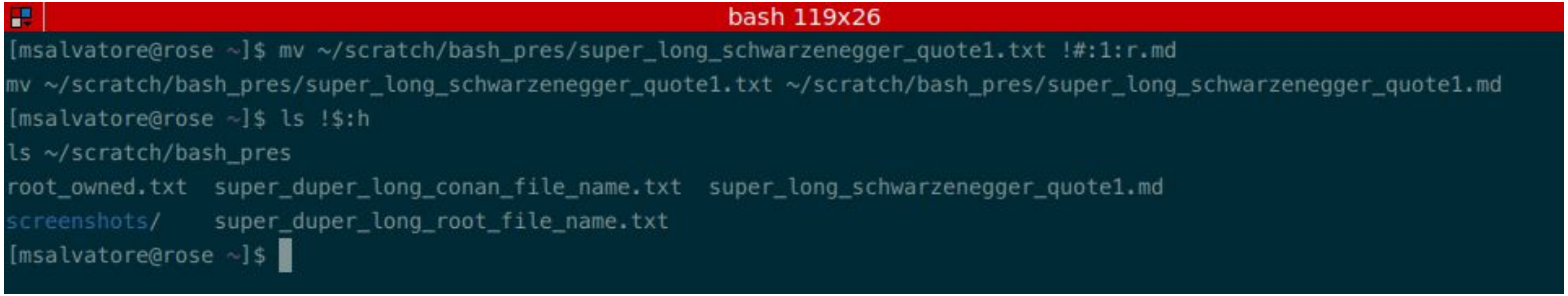


```
[msalvatore@rose ~]$ cat ~/scratch/bash_pres/super_long_schwarzenegger_quote1.txt
Put that cookie down!
[msalvatore@rose ~]$ ls -la !$:h
ls -la ~/scratch/bash_pres
total 32
drwxr-xr-x  3 msalvatore msalvatore 4096 Jun 17 12:10 ./
drwxrwx--- 15 msalvatore msalvatore 4096 Jun 17 10:04 ../
-rw-------  1 root       root         19 Jun 17 10:06 root_owned.txt
drwxrwxr-x  2 msalvatore msalvatore 4096 Jun 17 12:22 screenshots/
-rw-r-----  1 msalvatore msalvatore  125 Jun 17 11:21 super_duper_long_conan_file_name.txt
-rw-------  1 root       msalvatore   49 Jun 17 11:34 super_duper_long_root_file_name.txt
-rw-r-----  1 msalvatore msalvatore   22 Jun 17 12:10 super_long_schwarzenegger_quote1.txt
-rw-r-----  1 msalvatore msalvatore   22 Jun 17 12:14 super_long_schwarzenegger_quote2.txt
[msalvatore@rose ~]$ 
```

Using the History: Modifiers — r

Remove a trailing suffix of the form '.suffix', leaving the basename.

Note that the `!#` event designator refers to the current command, so `!#:1` means, "the first argument from the current command."



```
[msalvatore@rose ~]$ mv ~/scratch/bash_pres/super_long_schwarzenegger_quote1.txt !#:1:r.md
mv ~/scratch/bash_pres/super_long_schwarzenegger_quote1.txt ~/scratch/bash_pres/super_long_schwarzenegger_quote1.md
[msalvatore@rose ~]$ ls !$:h
ls ~/scratch/bash_pres
root_owned.txt  super_duper_long_conan_file_name.txt  super_long_schwarzenegger_quote1.md
screenshots/    super_duper_long_root_file_name.txt
[msalvatore@rose ~]$
```

Using the History: Modifiers — r

!# — Take the current command

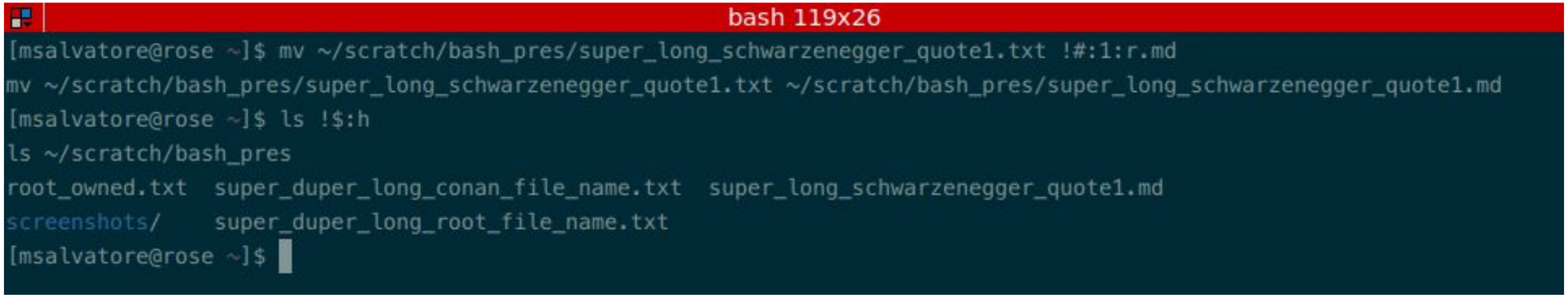
“mv ~/scratch/bash_pres/super_long_schwarzenegger_quote.txt”

!#:1 — Get the first argument from the current command

~/scratch/bash_pres/super_long_schwarzenegger_quote1.txt

!#:1:r — Strip the “.txt” suffix off of the file name

~/scratch/bash_pres/super_long_schwarzenegger_quote

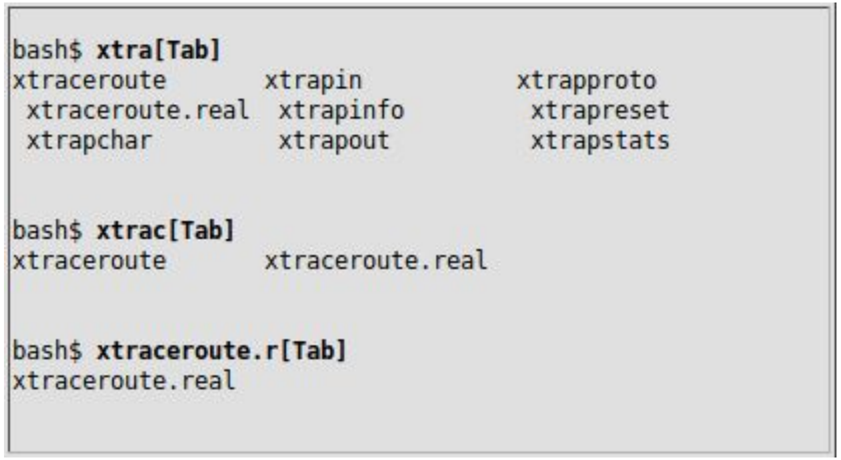


```
[msalvatore@rose ~]$ mv ~/scratch/bash_pres/super_long_schwarzenegger_quote1.txt !#:1:r.md
mv ~/scratch/bash_pres/super_long_schwarzenegger_quote1.txt ~/scratch/bash_pres/super_long_schwarzenegger_quote1.md
[msalvatore@rose ~]$ ls !$:h
ls ~/scratch/bash_pres
root_owned.txt  super_duper_long_conan_file_name.txt  super_long_schwarzenegger_quote1.md
screenshots/    super_duper_long_root_file_name.txt
[msalvatore@rose ~]$
```

Bash Keybindings and Other Shortcuts

Tab Completion

Tab Completion allows you to type the first part of a command and use the [Tab] key to auto-complete the command sequence. See https://www.tldp.org/LDP/abs/html/tabexpansion.html for more detail.



```
bash$ xtra[Tab]
xtraceroute       xtrapin           xtrapproto
 xtraceroute.real  xtrapinfo         xtrapreset
 xtrapchar         xtrapout          xtrapstats


bash$ xtrac[Tab]
xtraceroute       xtraceroute.real


bash$ xtraceroute.r[Tab]
xtraceroute.real
```

VI Mode

Bash has a vi mode that allows you to use vi keybindings to modify your commands

Simply type `set -o vi` to enable vi mode.

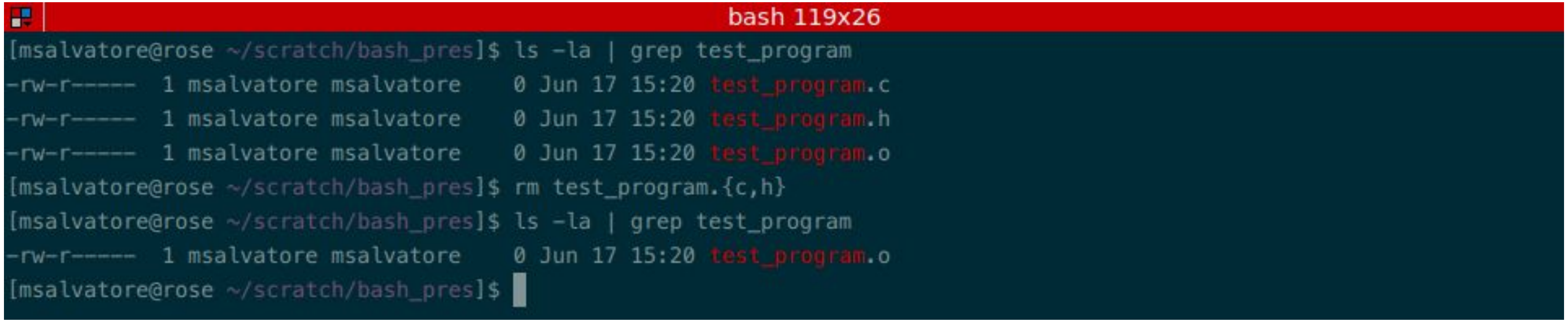
See https://sanctum.geek.nz/arabesque/vi-mode-in-bash/ for more details

Bash Keybindings

ctrl + _ (undo)
ctrl + arrow (move forward a word)
ctrl + a (move cursor to start)
ctrl + e (move cursor to end)
ctrl + k (cuts everything after the cursor)
ctrl + l (clears screen)
ctrl + q (resume command that is in the foreground)
ctrl + s (pause a long running command in the foreground)
ctrl + t (swap two characters)
ctrl + u (cuts everything before the cursor)
ctrl + x + ctrl + e (opens the command string in an editor so that you can edit it before it runs)
ctrl + x + * (expand glob/star)
ctrl + xx (move to the opposite end of the line)
ctrl + y (pastes from the buffer)
ctrl + shift + c/v (copy/paste into terminal)

Brace Expansion

Brace expansion allows you to generate arbitrary strings/permutations.



```
[msalvatore@rose ~/scratch/bash_pres]$ ls -la | grep test_program
-rw-r-----  1 msalvatore msalvatore    0 Jun 17 15:20 test_program.c
-rw-r-----  1 msalvatore msalvatore    0 Jun 17 15:20 test_program.h
-rw-r-----  1 msalvatore msalvatore    0 Jun 17 15:20 test_program.o
[msalvatore@rose ~/scratch/bash_pres]$ rm test_program.{c,h}
[msalvatore@rose ~/scratch/bash_pres]$ ls -la | grep test_program
-rw-r-----  1 msalvatore msalvatore    0 Jun 17 15:20 test_program.o
[msalvatore@rose ~/scratch/bash_pres]$
```

rm test_program.{c,h} -> rm test_program.c test_program.h

Making the Most of cd

- $> cd   # Takes you to your home directory
- $> cd .. # Takes you up one level in the directory tree
- $> cd - # Takes you back to the previous directory

Bash keeps a list of remembered directories called the “directory stack”. You can use the commands `dirs`, `popd`, and `pushd` to manipulate the directory stack. See https://www.gnu.org/software/bash/manual/html_node/Directory-Stack-Builtins.html#Directory-Stack-Builtins for more information.

Using cd with Environment Variables

Environment variables are simply variables. They are available to the shell and its subprocesses.

```
$> pwd
/tmp
$> export DL=/home/usr/Downloads
$> cd $DL
$> pwd
/home/usr/Downloads
$>
```

Scripts, Aliases, and Functions

Scripts

A script is a set of commands contained within a text file. You can script anything you can type on the command line. Scripts help you type less by allowing you to invoke a set of commands by simply executing the script.

The first line of any bash script should be “#!/bin/bash”. This may vary based on the location of your bash executable, but the shebang “#!” remains the same.

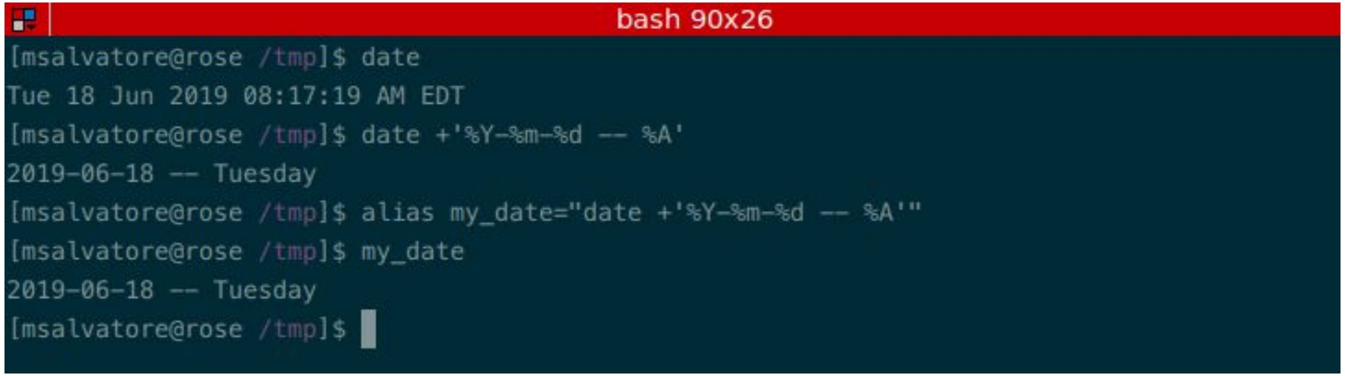
Example Script

This script is called “backup.sh”. I use it to backup my personal files to a remote backup server.

```
#!/bin/bash

echo “starting backup...”

rsync --exclude="remote" --exclude=".cache/google-chrome" -avz -e "ssh -i ~/.ssh/id_user_backup" --delete /home/user \
backup_server:/backup_directory

rsync --exclude="lost+found" --exclude home/user/Downloads --exclude media_backup -avz -e "ssh -i ~/.ssh/id_user_backup" \
--delete /data/disk1 backup_server:/backup_directory

rsync -avz -e "ssh -i ~/.ssh/id_user_backup" --delete /data/disk2 backup_server:/backup_directory

echo “finished backup...”
```

Aliases

Put simply, a bash alias is an abbreviation or nickname for another command. You define aliases using the “alias” builtin.



```
[msalvatore@rose /tmp]$ date
Tue 18 Jun 2019 08:17:19 AM EDT
[msalvatore@rose /tmp]$ date +'%Y-%m-%d -- %A'
2019-06-18 -- Tuesday
[msalvatore@rose /tmp]$ alias my_date="date +'%Y-%m-%d -- %A'"
[msalvatore@rose /tmp]$ my_date
2019-06-18 -- Tuesday
[msalvatore@rose /tmp]$
```

Favorite Alias

I have a lot of aliases. By far, this one is my favorite:

```
$> alias emacs=vim
```

Complex commands and aliases with ;, &&, and ||

Aliases (or regular command-line commands) can be made more complex by using the ";", "&&", and "||" operators.

- ; -- signifies the end of a command
- && -- Logical "and" -- Execute the following command only if the first command was successful
- || -- Logical "or" -- Execute the following command only if the first command failed

Example:

- $> alias lscd="ls -la .; cd $HOME/Downloads"
  - # List the contents of the current directory and cd to $HOME/Downloads
- $> alias mkrun="make && ./build/bin/run"
  - # Run "make" to compile the project. If compilation is successful, run the program
- $> alias mkcl="make || make clean"
  - # Run "make" to compile the project. If compilation fails, run "make clean" to cleanup

Functions

Functions can be declared and used from within a script or used directly from the command line. For the sake of brevity, I’ll not discuss functions. For more information on calling functions from the command line, see https://superuser.com/questions/106272/how-to-call-bash-functions

Storing Variables and Aliases for Later Use

The ~/.bashrc file gets sourced every time you start bash. Adding aliases and variables to this file will make them available to you in every new bash instance.

Security Implications

The PATH environment variable

The PATH environment variable contains a list of directories that contain executables. When you invoke any executable on the command line, bash searches the directories listed in your PATH to find the executable.

An improperly configured PATH (or one modified by an attacker) can trick you into executing malicious code.

Live Demo

Thank you. Questions?

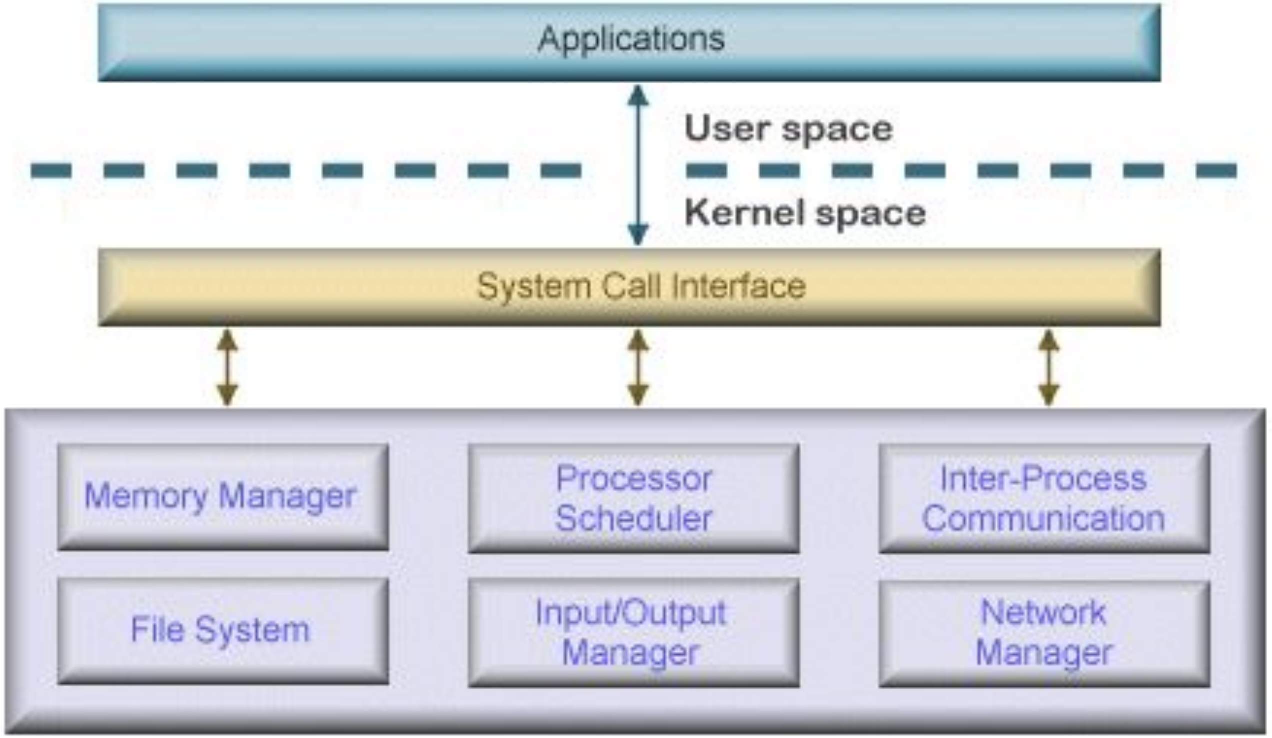
Backup

Intro to strace

“Strace is a useful diagnostic, instructional, and debugging tool.” It’s also a super cool red and blue team tool.

Strace traces all system calls made by a process. Almost everything interesting that a process does involves a system call. System calls allow processes in user space to perform such actions as allocating memory, accessing files, sending/receiving network traffic, load kernel modules, gather input from HIDs, and more!

System Call Diagram



Some strace Uses

- Debugging
- Education
- Defensive or offensive binary analysis and inspection