RIT | Golisano College of Computing and Information Sciences
School of Information

# NSSA221 - System Administration I

## Scripting Assignment 03 – Symbolic Links

**The Basics:**

Symbolic links are widespread in Linux systems. They allow easy access to files or maintain different versions of the same library. In many ways, we can think of symbolic links as being similar to shortcuts in Windows.

In this scripting exercise, you will become familiar with the various commands used to find and create symbolic links and their associated target paths.   Bash and Python commands are available and may be used in the script. For example, in Bash the command to create a symbolic link is `ln -s`, the equivalent in Python is the `os.symlink` method. Make sure to use the man pages and Python documentation to find the syntax needed for the various commands.

**Script Requirements:**

**The script must be written in Python 3 and titled *"sym_link.py."* It will be run and tested on the CentOS 8 virtual machine in your infrastructure.** The infrastructure is the enterprise environment you are working in; your supervisor does not care that it can run on your laptop. The instructor or teaching assistant will run it and award points based on the requirements and overall functionality. Points awarded will be based on the criteria in *"Table 1 – Script Grading Rubric."*

**Additional Information:**

This script aims to create symbolic links in the user's home directory and only in that directory. We often find that we have broken links elsewhere in the system, causing instability and odd behavior. In Windows, most users know that to create a shortcut, right-click on the file and select *"Create Shortcut."* This script should do something similar. The end-user does not want to be bothered with learning Linux "commands," so we want to create a script that will simplify the process for them. Like the ping test, create a menu with the options to make a symbolic link, delete a link, and run an easily readable report summarizing the symbolic links in the user's home directory. Please refer to the scripting requirements table for more details and the grading rubric.

Script Demo:

To view the script in action, run the script command.

```
scriptreplay --timing=time.log shortcut.scr
```

**Table 1** – Script Grading Rubric

| **Requirements** | **Points** | **Points Earned** |
|---|---|---|
| Script contains the shebang! | 2 | |
| Script has executable permission set. | 2 | |
| Script is commented with student's name, date. | 2 | |
| **Script is titled "*shortcut.py*"** | 2 | |
| Script clears the terminal when it runs. | 2 | |
| User is informed of their current working directory. | 5 | |
| Menu is designed for end user readability. | 5 | |
| User is prompted to enter the file that they want to enter a short for. | 5 | |
| Scripts checks to see if the file exits, and user is informed if it does not. | 5 | |
| **The script uses the following Linux or associative Python commands; `find`, `readlink`, and `ln`.** | 10 | |
| **The script uses the subprocess, or os modules.** | 5 | |
| **The user exists the script by typing "*quit*."** | 5 | |
| The summary report lists the symbolic links in the user's home directory. | 5 | |
| The summary report shows the number of links in the user's home directory. | 5 | |
| The summary report shows the target path for each link. | 5 | |
| The report is in a user-friendly readable format. | 5 | |
| The script is sufficiently commented. | 5 | |
| The script is written in Pythonic style. | 5 | |
| Scripts runs with no errors. | 10 | |
| Script is fully functional and runs as expected. | 10 | |
| **Final Grade** | | |