

## Assignment #3—Off The Beaten \$PATH

---

**Due: February 6, 2023 at 11:59pm**

This assignment consists of three different components:

1. **You will modify your \$PATH variable, shell prompt, and add aliases**
2. **You'll get some practice using the networking tools we learned about in Lecture 7**
3. **You'll run a small server that our grading machine will connect to**

We expect this assignment to take 1-3 hours depending on your proficiency level with the tools. If you find yourself unproductively stuck or unproductively struggling, ask on Ed and/or go to office hours!

### **Part I: Customizing Your Environment Variables (1 point)**

In Lecture 4: Shell Scripting, we saw how to write a shell script and make it executable from anywhere on the computer. Imagine you have a shell script called `hello.sh` that simply prints `Hello`. You create the script inside of a folder called `CS45` on your Desktop. (The folder would thus have a path of `~/Desktop/CS45`.) Given the script is located inside of the `CS45` folder, you can only run it from within that folder. Let's change that!

To make `hello.sh` script executable from anywhere on your machine, you will want to move it somewhere that is recognized by your `$PATH` environment variable. Whenever you type a command in the shell prompt (i.e. the command `grep`), your computer searches every folder inside of your `$PATH` environment variable to see if any of those folders have an executable by the name of the command you typed in (i.e. an executable by the name `grep`). For example, here is a sample `$PATH` environment variable:

Unset

```
/Library/Frameworks/Python.framework/Versions/3.11/bin:/opt/local/bin:/opt/local/sbin:/opt/homebrew/bin:/opt/homebrew/sbin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Library/Apple/usr/bin
```

When you type in `grep`, your computer first searches inside of `/Library/Frameworks/Python.framework/Versions/3.11/bin` to see if it finds an executable by the name of `grep`. If it doesn't find it there, it would then search inside of `/opt/local/bin`. It would then search inside of `/opt/local/bin`, and so forth.

To check what paths are currently set on your computer, you can run `echo $PATH` at the command line prompt.

Let's create a new `bin` folder that belongs to the user who is currently signed in (presumably you!) A `bin` folder is short for a binary folder, which, as the name suggests, contains binary files

(executable files) for programs that you might want to run. This new `bin` folder will allow us to store any of the local binary files that should be accessible anywhere on the computer for the current user.

### Step 1: Creating A Bin Folder

First, you will want to navigate to your home directory using `cd ~`. Once you are in your home directory, you will want to make a new folder called `bin` using `mkdir bin`. Next, you will want to enter this new directory using `cd`. We will need the absolute path of this directory for Step 3. To get the absolute path of your newly-created `bin` folder, you should run `pwd` inside `~/bin`.

### Step 2: Finding Your Shell

Your next task is to find out what shell you are running. You can normally do this by running `ps -p $$`. Your output may look something like the following:

Unset

```
karel@karel-computer bin % ps -p $$
  PID TTY          TIME CMD
 25466 ttys001    0:00.32 -zsh
```

### Step 3: Adding the Path

Now that you have your local `bin` folder (`~/bin`) and your shell, you will need to update (or make!) the config file specific to your shell:

- If you are using a `zsh` shell, your config file will be `.zshrc`
- If you are using a `bash` shell, your config file will be `.bashrc`
- If you are using a `tcsh` shell, your config file will be `.tcshrc`

To check if you already have an existing config file, you should navigate back to your home directory (`~`) and then run `ls -a` (which will list all hidden files, such as your configuration files). If you don't have the right configuration file for your shell, you can just create the file using `touch <NAME_OF_FILE>` (i.e. `touch .zshrc` for a `zsh` shell).

Once you have your configuration file, you will want to open it and the following line, replacing `NEW_PATH` with the full path you discovered in Step 1 using `pwd`:

Unset

```
export PATH=$PATH:<INSERT_NEW_PATH>
```

In other words, if your path from step 1 was `/Users/karel/bin`, then your line inside of your config file would read `export PATH=$PATH:/Users/karel/bin`

Congratulations! You've successfully added a new path to your environment. You should now test out your new powers by creating a script called `hello` (you can drop the `.sh` ending as we will be turning the script into a command). The script should simply echo `Hello` along with your name. Make sure it has a valid shebang line so your computer knows how to run it! You can create the script in any directory you choose. Once you have created your script, turn it into an executable by running `chmod +x hello`

Now you will want to move the script to `~/bin`. You can do so by running `mv hello ~/bin/`. You should now be able to say hello to yourself at any time of day, from anywhere on the computer!

In addition to modifying your path variable, there are other useful configurations that we will guide you through. Let's work on implementing the following two configurations:

- Adding colors to your `ls` command
- Customizing your shell prompt

### *Adding Colors to ls*

To add colors for `ls`, you will want to add an alias for `ls` that changes the standard `ls` command to an `ls` command with colors. The way to do this will depend on which shell you are using.

If you are on macOS, you should add the following two lines to your `.bashrc` or `.zshrc` file:

```
Unset
export CLICOLOR=1
alias ls='ls -G'
```

If you are using Linux or Windows (WSL), you should add the following line to your `.bashrc` or `.zshrc` file:

```
Unset
alias ls="ls --color=auto"
```

### *Customizing Your Shell Prompt*

Now we will customize our shell prompt by adding colors to it and modifying what contents it has. Though we will leave it up to you as to what customization you would like to include, we have also provided a sample customization with CS45 themed-colors.

If you are using a `zsh` shell, you should use `zsh` guidelines for customizing your prompt. [Here](#) is a tool to build a `zsh` prompt. You can also use the chart below to choose `zsh` colors.

000000	005f00	008700	00af00	00d700	00ff00	5fff00	5fd700	5faf00	5f8700	5f5f00	5f0000
016	022	028	034	040	046	082	076	070	064	058	052
00005f	005f5f	00875f	00af5f	00d75f	00ff5f	5fff5f	5fd75f	5faf5f	5f875f	5f5f5f	5f005f
017	023	029	035	041	047	083	077	071	065	059	053
000087	005f87	008787	00af87	00d787	00ff87	5fff87	5fd787	5faf87	5f8787	5f5f87	5f0087
018	024	030	036	042	048	084	078	072	066	060	054
0000af	005faf	0087af	00afaf	00d7af	00ffaf	5fffaf	5fd7af	5fafaf	5f87af	5f5faf	5f00af
019	025	031	037	043	049	085	079	073	067	061	055
0000d7	005fd7	0087d7	00afd7	00d7d7	00ffd7	5fffd7	5fd7d7	5fafd7	5f87d7	5f5fd7	5f00d7
020	026	032	038	044	050	086	080	074	068	062	056
0000ff	005fff	0087ff	00afff	00d7ff	00ffff	5fffff	5fd7ff	5fafff	5f87ff	5f5fff	5f00ff
021	027	033	039	045	051	087	081	075	069	063	057
8700ff	875fff	8787ff	87afff	87d7ff	87ffff	afffff	afd7ff	afafff	af87ff	af5fff	af00ff
093	099	105	111	117	123	159	153	147	141	135	129
8700d7	875fd7	8787d7	87afd7	87d7d7	87ffd7	afffd7	afd7d7	afafd7	af87d7	af5fd7	af00d7
092	098	104	110	116	122	158	152	146	140	134	128
8700af	875faf	8787af	87afaf	87d7af	87ffaf	afffaf	afd7af	afafaf	af87af	af5faf	af00af
091	097	103	109	115	121	157	151	145	139	133	127
870087	875f87	878787	87af87	87d787	87ff87	afff87	afd787	afaf87	af8787	af5f87	af0087
090	096	102	108	114	120	156	150	144	138	132	126
87005f	875f5f	87875f	87af5f	87d75f	87ff5f	afff5f	afd75f	afaf5f	af875f	af5f5f	af005f
089	095	101	107	113	119	155	149	143	137	131	125
870000	875f00	878700	87af00	87d700	87ff00	afff00	afd700	afaf00	af8700	af5f00	af0000
088	094	100	106	112	118	154	148	142	136	130	124

If you'd like to use our CS45-themed shell prompt, you should add the following line to your `.zshrc` file:

Unset

```
PROMPT='%B%F{75}%n%f@%b%F{88}%m%f:%~ %# '
```

If you are using a `bash` shell, you can also customize your shell. You will need to use `bash` specific syntax. You can easily customize your prompt using [this tool](#). If you'd like to use our CS45-themed shell prompt, you should add the following line to your `.bashrc` file:

Unset

```
PS1="\[$(tput bold)\]\[\033[38;5;75m\]\u\[$(tput sgr0)\]@\[$(tput sgr0)\]\[\033[38;5;88m\]\h\[$(tput sgr0)\]: \w \\\$ \[$(tput sgr0)\] "
```

For students who have access to the myth machines, we also recommend adding an alias for `ssh`-ing into the myth machines. (This part is not graded as not all students have access to the myth machines.) While you won't be able to have it automatically enter your password into `ssh` (for security reasons, `ssh` requires a human to type in the password), it'll at least reduce the tedium of typing "`ssh $SUNET@myth.stanford.edu`" over and over again.

**Note:** If you are using another shell and we didn't include specific instructions here, reach out to us! We are happy to help.

For this part of the assignment, you should submit your configuration file (e.g. `.bashrc`, `.zshrc`, etc).

## Part II: Networking Short Answers (1 point)

In this part of the assignment, you'll be exploring some of the networking tools we learned about in Lecture 7 to get some information about your computer's networking environment. Make sure to install [the software for Lecture 7](#)! Note that for the commands below, if your computer uses Windows, you should use the Windows commands (even if you're inside WSL!)

### (1) Network Interfaces & IP Addresses

To start, let's take a look at what network interfaces your computer has. On Windows, you can run the command `ipconfig.exe`, on macOS you can use `ifconfig`, and on linux you can use `ip addr`. This will list all the network devices your computer has available!

1.1 Using the command above, redirect its output into a file called `interfaces.txt`.

1.2 Look inside the output in `interfaces.txt`. Write the name of the interface that appears to be your wireless connection into a file called `wifi_interface.txt`

1.3 Look inside the output of `interfaces.txt`. Find and write your local IP address inside a file called `Local_ip.txt`.

### (2) Routes

Let's take a look at the routing table your computer is using. On Windows, you can run the command `route.exe print -4`, on macOS you can use `netstat -nrf inet`, and on linux you can use `ip -4 route`

2.1 Using the command above, redirect its output into a file called `routes.txt`

2.2 Look inside the output in `routes.txt`. Find the default route (this is the address of the router that's connecting you to the rest of the internet!) and put its IP address in `default_route.txt`.

### (3) Traceroute

Let's see what path it takes to get to a server hosted in another country (in this case, we'll be connecting to a website that gives information about a town in Japan)! On Windows, you can use the command `tracert.exe www.town.okutama.tokyo.jp`, on Linux you can use `traceroute -I --resolve-hostnames www.town.okutama.tokyo.jp`,<sup>1</sup> and on macOS you can use `traceroute -I www.town.okutama.tokyo.jp` Note that these commands may take a while to complete. If you have trouble with these commands, please let us know on Ed as soon as possible!

3.1 Using the commands above, pass them to a special program called `tee` which lets you redirect output to a file **and see it on your terminal at the same time!** The output of your command should go to a file called `traceroute.txt`. For example:

```
tracert.exe www.town.okutama.tokyo.jp / tee traceroute.txt
traceroute -I --resolve-hostnames www.town.okutama.tokyo.jp / tee traceroute.txt
traceroute -I www.town.okutama.tokyo.jp / tee traceroute.txt
```

3.2 How many hops did it take to get to the destination server? Put the number into a file `hops.txt`, or write "the traceroute didn't complete" if it wasn't able to find the destination within 64 hops.

3.3 Which hop number do you think was the last hop inside Stanford's campus? Put the number in `Last_stanford_hop.txt`

3.4 Which hop do you think is the first server you see that's located in Japan (if any)? Place your answer and justification in `jump.txt`

### Part III: Running a Small Server (1 point)

In Lecture 7: Introduction to Computer Networking, we learned all about how information travels from one computer to another. In this part of the assignment, you'll get some practice running your own development server over the network. Make sure to install [the software for Lecture 7!](#)



Note that this part of the assignment will expose parts of your computer to the internet. Please ensure that you follow the commands below in a **new, empty directory** to avoid exposing unwanted or private information.



<sup>1</sup> This is for the `inetutils-traceroute` package, which is the version listed on the software page. There's a few different programs named `traceroute`, so make sure you're using the right one!

In a **new directory**, create a new file called `sunet.txt` that contains your SUNet ID (the part before your email!):

```
Unset
mkdir my_server_directory
cd my_server_directory
echo "Your SUNet Here" > sunet.txt
```

Then, you'll want to start your server. First, open a Python server as before:

```
Unset
python3 -m http.server --bind localhost 8080
```

We want to also publish the server to the internet using `localtunnel`, so you'll need to **put your Python server into the background**. To do this, press **control+z** to suspend the process, then use **bg** to start it again in the background.

Finally, let's publish it on `localtunnel`. Make sure to [install it first](#)!

```
Unset
lt -p 8080 &
```

The **&** at the end of the command above instructs the shell to **immediately place localtunnel into the background** without needing to suspend it first. Neat!

Create a file called `server_url.txt` and copy/paste the URL that `localtunnel` gives you into the file. Then, submit it to Gradescope (keeping your computer open, `localtunnel` running, and Python HTTP server running)—our autograder will connect to your server and verify that your `sunet.txt` file matches your SUNet in Gradescope.

If you need to resubmit your assignment, make sure the server is running and the URL in `server_url.txt` is up-to-date—otherwise our autograder won't be able to connect to your computer.

### **Feedback Survey (0.5 points)**

Once you have completed the assignment, you can earn an additional 0.5 points by completing our anonymous feedback survey. Given this is the first offering of the course, we want to collect as much feedback as possible to improve the course in the future. You can complete the survey [here](#). Once you complete the survey, you will receive a completion code which you should place in a text file named `survey.txt`. The survey is anonymous so submitting the completion code is the only

way to verify that you completed the survey. *Please do not share this code with anyone, as that would constitute a breach of the honor code.*

### Submitting Your Assignment

Once you have finished this assignment, you will need to upload your files to [Gradescope](#). Make sure to upload all files to the Assignment 3 submission page. You should also upload `survey.txt` if you completed the survey.

You can submit all necessary files by running the following command, replacing `<CONFIG_FILE>` with the configuration file for your shell.

Unset

# Run this command in your assignment directory:

```
zip -jv assign3_submission.zip ./server_url.txt ./survey.txt
./interfaces.txt ./wifi_interface.txt ./local_ip.txt ./routes.txt
./default_route.txt ./traceroute.txt ./hops.txt ./last_stanford_hop.txt
./jump.txt ~/<CONFIG_FILE>
```

Once you have created a zip file, you can upload it to Gradescope. Make sure your server is running and available at the URL specified in the `server_url.txt` file while the autograder is running.

*All files must have the same name as specified above.*