MERCK
INVENTING FOR LIFE

The Linux commands to facilitate programming workflow

Peikun Wu

Merck & Co., Inc., Kenilworth, NJ, USA 2021

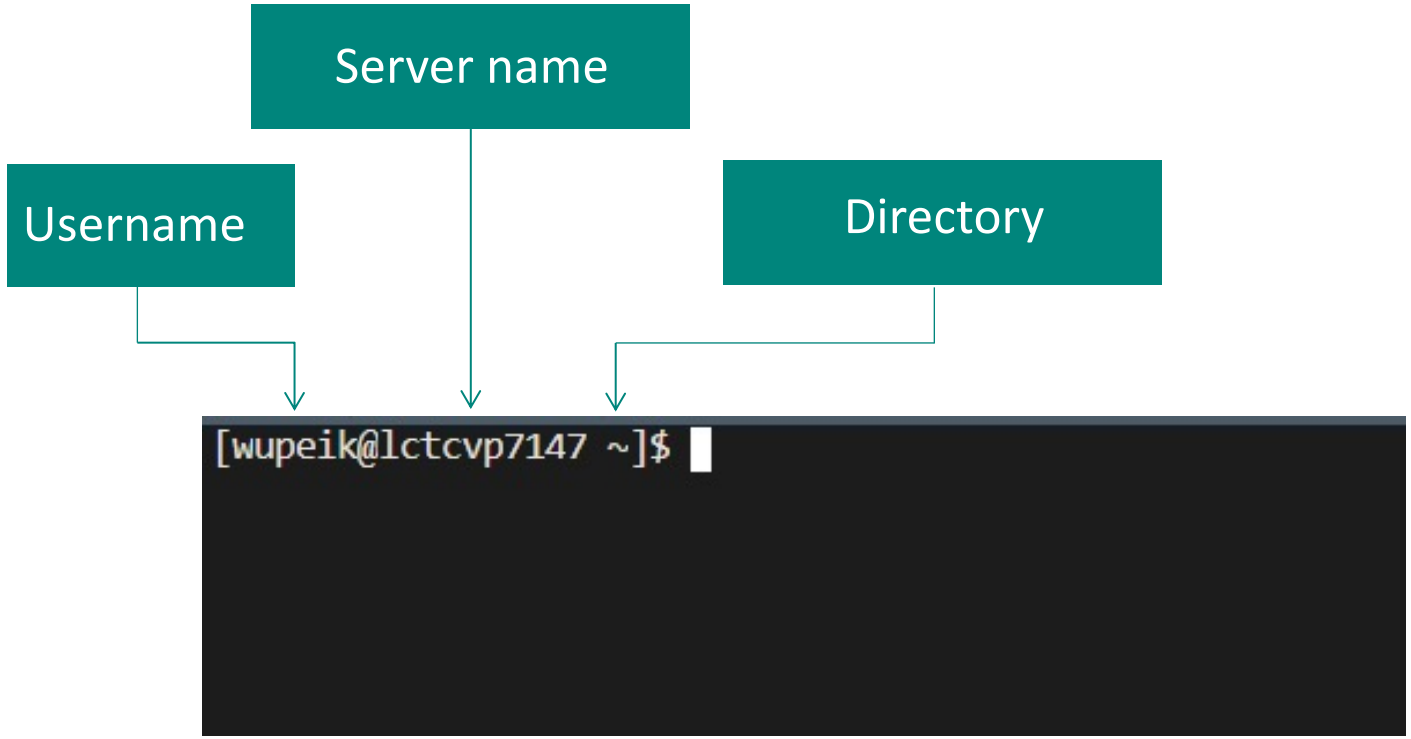
October, 2021

Overview

1. The shell and common shell commands
2. Variables, tests
3. Shell scripting
4. Script examples

The shell

- What is the shell?
- What is bash?
- What is a terminal



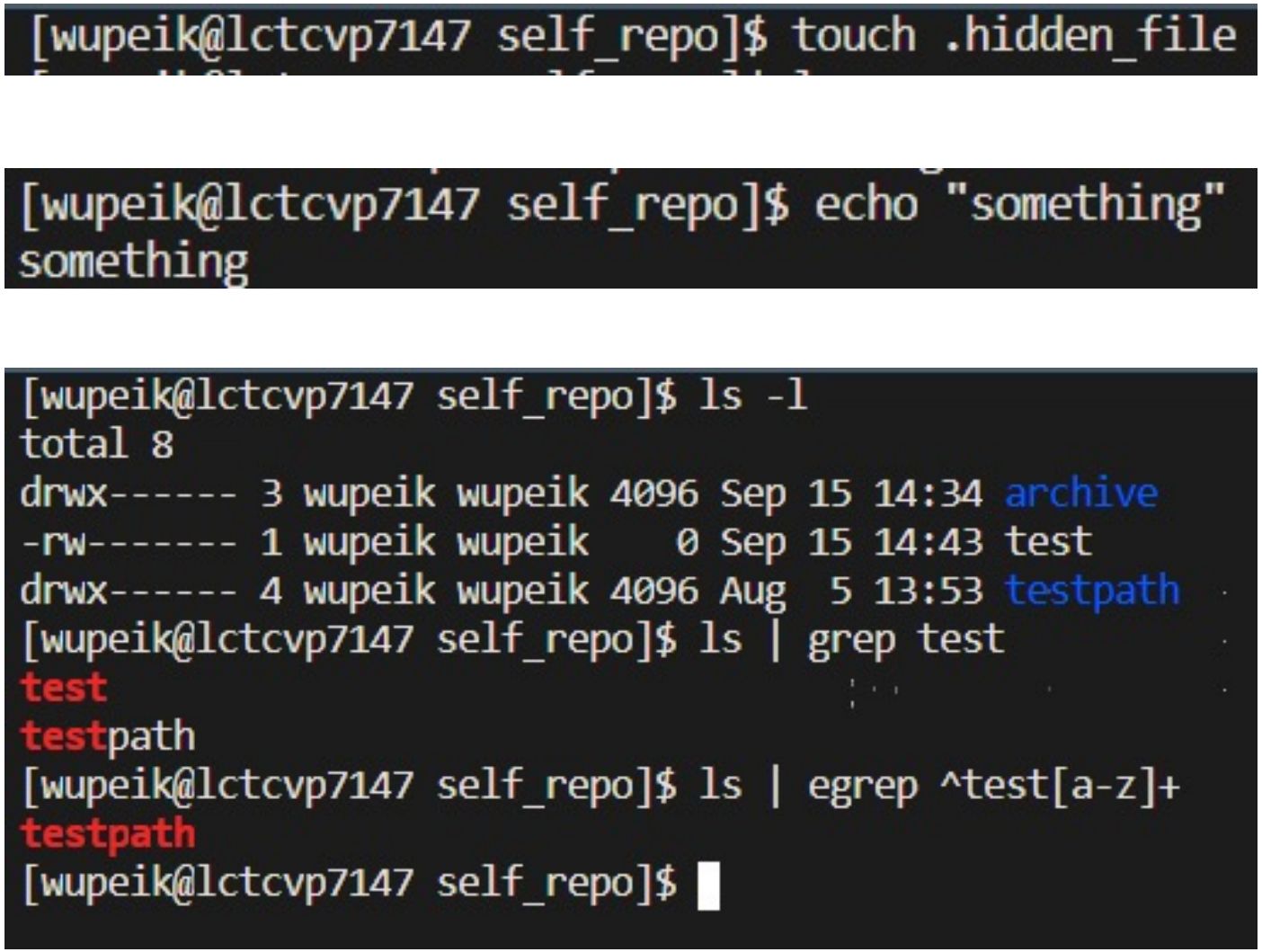
Common commands

- cd
- ls
  - ls -l , ls -la

```
[wupeik@lctcvp7147 self_repo]$ ls
archive  testpath
[wupeik@lctcvp7147 self_repo]$ ls -l
total 8
drwx------ 3 wupeik wupeik 4096 Sep 15 14:34 archive
drwx------ 4 wupeik wupeik 4096 Aug  5 13:53 testpath
[wupeik@lctcvp7147 self_repo]$ ls -la
total 16
drwx------  4 wupeik wupeik 4096 Sep 15 14:35 .
drwx------ 21 wupeik wupeik 4096 Aug 18 11:34 ..
drwx------  3 wupeik wupeik 4096 Sep 15 14:34 archive
-rw-------  1 wupeik wupeik    0 Sep 15 14:35 .hidden_file
drwx------  4 wupeik wupeik 4096 Aug  5 13:53 testpath
```

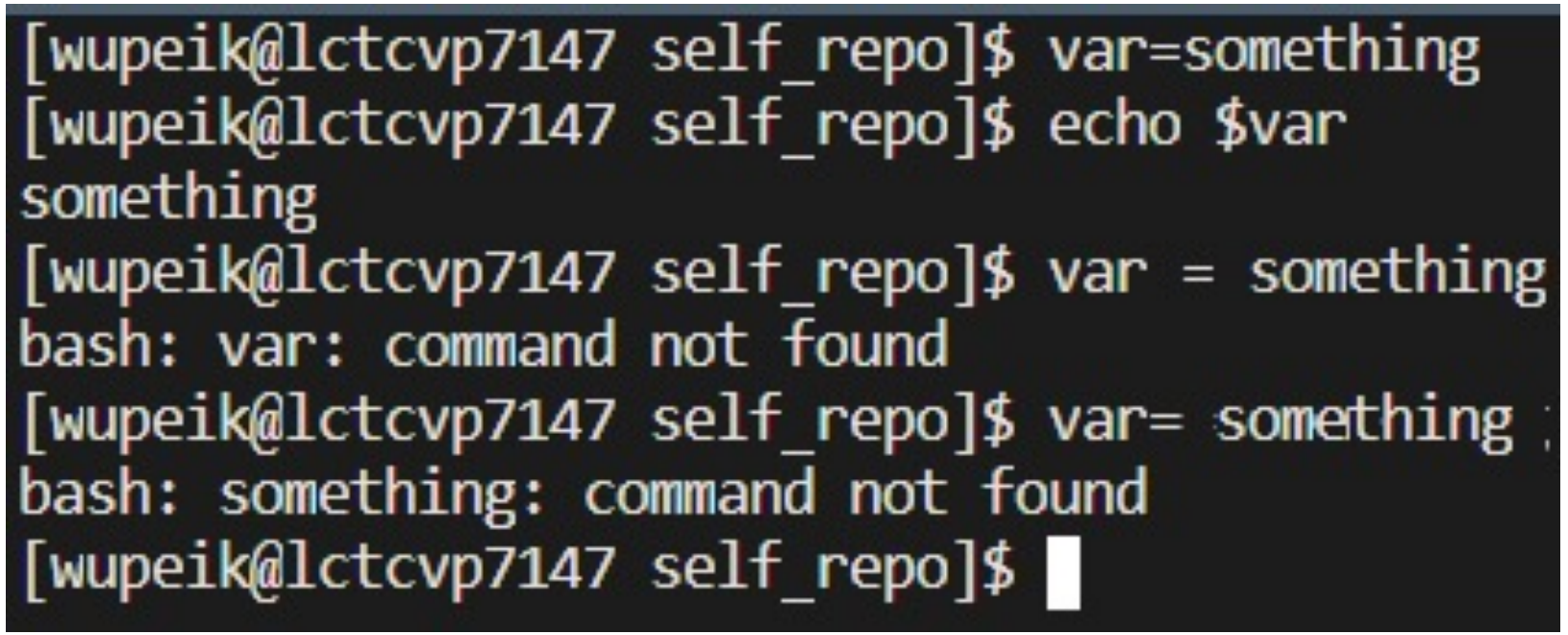
Common commands

- mkdir
- touch
- echo
- cp, mv
- grep
- find



Variables

no space



Variables

```
[wupeik@lctcvp7147 self_repo]$ dir=$(pwd)
[wupeik@lctcvp7147 self_repo]$ echo $dir
/home/wupeik/self_repo
[wupeik@lctcvp7147 self_repo]$
```

What if

```
[wupeik@lctcvp7147 self_repo]$ if [[ "$dir" == $(pwd) ]] ; then echo "true"; fi
true
```

```
1  if [[ $dir == $(pwd) ]]; then
2      echo "true"
3  fi
4
```


send to terminal

What if

integer comparison:

- -gt
- -lt
- -ge
- -le
- -eq
- -ne

conditional evaluation:

- &&
- ||

files and directories:

- -e
- -d

What if

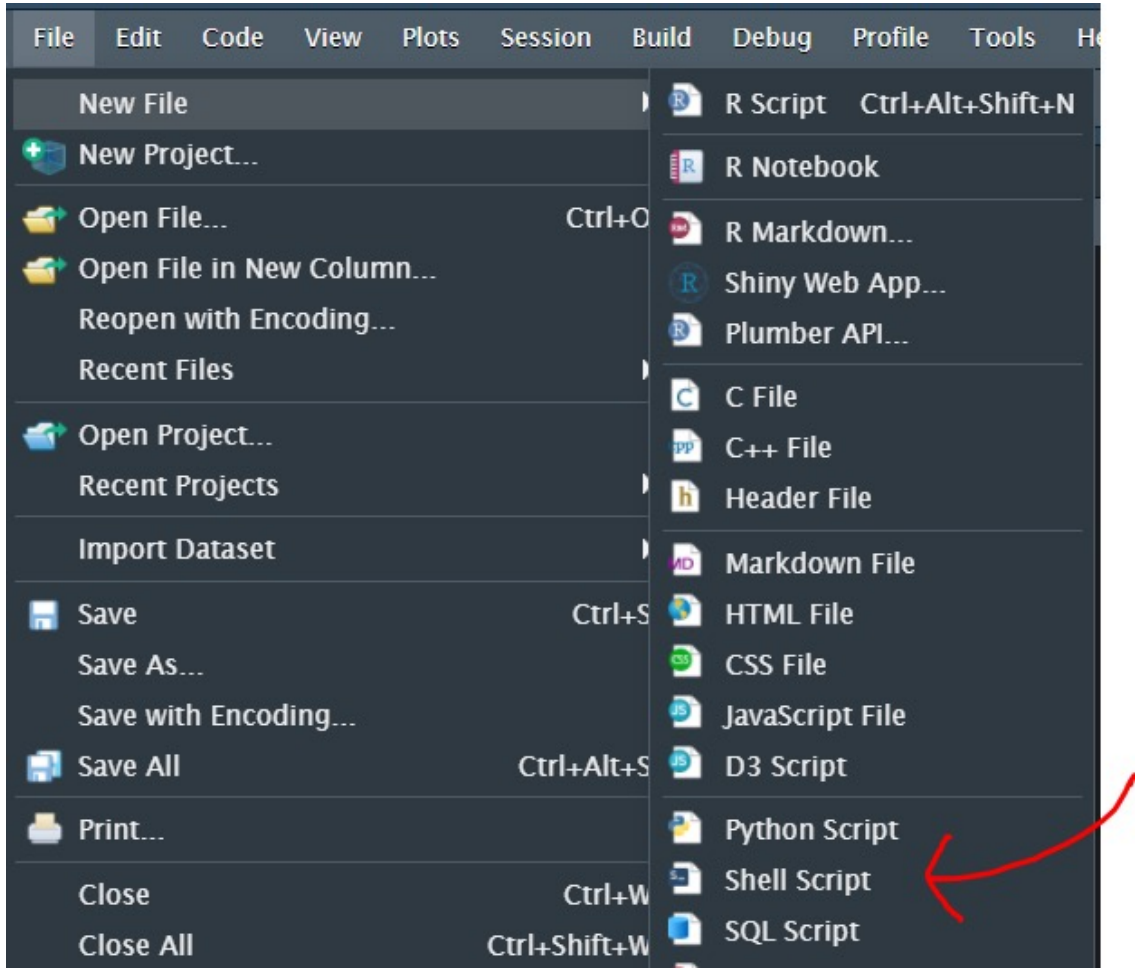
```
[wupeik@lctcvp7147 self_repo]$ true && echo "true"
true
[wupeik@lctcvp7147 self_repo]$ true || echo "true"
[wupeik@lctcvp7147 self_repo]$ false || echo "true"
true
```



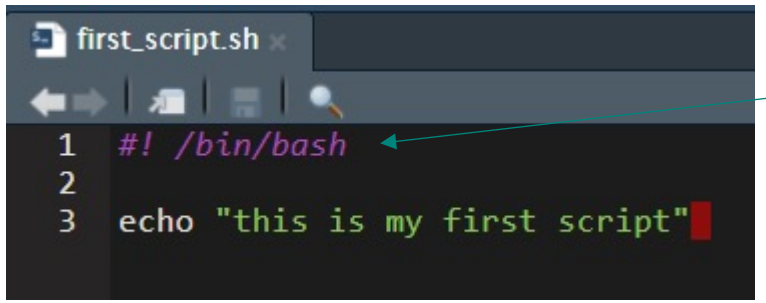
```
[wupeik@lctcvp7147 self_repo]$ [[ $dir == $(pwd) ]] && echo "true" || echo "false"
true
[wupeik@lctcvp7147 self_repo]$ [[ $dir != $(pwd) ]] && echo "true" || echo "false"
false
```

Shell Scripting

You can develop shell scripts in RStudio



Shell Scripting



Shebang: indicate an interpreter for execution under UNIX / Linux operating systems

```
[wupeik@lctcvp7147 ~]$ cd self_repo
[wupeik@lctcvp7147 self_repo]$ ls
archive  first_script.sh  test  testpath
[wupeik@lctcvp7147 self_repo]$ ./first_script.sh
bash: ./first_script.sh: Permission denied
[wupeik@lctcvp7147 self_repo]$ chmod +x ./first_script.sh
[wupeik@lctcvp7147 self_repo]$ ./first_script.sh
this is my first script
[wupeik@lctcvp7147 self_repo]$
```

Shell Scripting

first_script.sh

```
1  #! /bin/bash
2
3  echo $1
```

- $0: Script's name
- $1 - $9: Arguments
- $?: Return code from the previous command

```
[wupeik@lctcvp7147 self_repo]$ ./first_script.sh hello
hello
[wupeik@lctcvp7147 self_repo]$ echo $?
0
```

Example: setup validation directories

- adamdata
- sdtmdata
- program
- validate
- zzz_pw

- program1
- program2
- program3
- …

- dbl-pgm/input
- dbl-pgm/output
- dbl-pgm/pgm

object: to create multiple folders at once

use mkdir pgmname/folder1
mkdir pgmname/folder2
…

Example: setup validation directories

```
#! /bin/bash
# script to create validation folder
# input 1 : study path
# input 2 : program name

loc=$1/validate

if [[ ! -d $loc/$2/dbl-pgm ]]
  then mkdir -p $loc/$2/dbl-pgm/{input,output,pgm}
    cp $1/program/startup.sas $loc/$2/dbl-pgm/pgm
    echo "validation folder created successfully"
else
  echo "validation forlder already exists"
fi
```

```
[wupeik@lctcvp7147 zzz_pw]$ ./val_dir.sh ~/self_repo/testpath/mk123 program1
validation folder created successfully
[wupeik@lctcvp7147 zzz_pw]$ ls ../validate -R
../validate:
program1

../validate/program1:
dbl-pgm

../validate/program1/dbl-pgm:
input  output  pgm

../validate/program1/dbl-pgm/input:

../validate/program1/dbl-pgm/output:

../validate/program1/dbl-pgm/pgm:
startup.sas
[wupeik@lctcvp7147 zzz_pw]$ ./val_dir.sh ~/self_repo/testpath/mk123 program1
validation forlder already exists
[wupeik@lctcvp7147 zzz_pw]$
```

Example: setup validation directories

```
[wupeik@lctcvp7147 zzz_pw]$ ls ../program
adae.sas  adsl.sas  startup.sas
[wupeik@lctcvp7147 zzz_pw]$
```

```
[wupeik@lctcvp7266 zzz_pw]$ ls ~/self_repo/testpath/mk123/program | grep .sas
adae.sas
adsl.sas
startup.sas
[wupeik@lctcvp7266 zzz_pw]$ ls ~/self_repo/testpath/mk123/program | grep .sas | grep -v startup
adae.sas
adsl.sas
```

```
for var in `ls ~/self_repo/testpath/mk123/program | grep .sas | grep -v startup`
    do ./val_dir.sh $dir $var
done
```

Example: prepare startup file

Original

```
%let prot  = /home/wupeik/self_repo/testpath/mk123;
libname outlib "somevalue"
filename outfile "somevalue";
```

Modified

```
%let prot  = /home/wupeik/self_repo/testpath/mk123;
%let protpath = &prot/newfolder;
libname outlib "&protpath"
filename outfile "somevalue";
```

- modify values for &prot
- add a new line after “%let prot = …”
- modify values for library outlib

find and replace

Example: prepare startup file

Use of Regex:

- ‘^’: starts with (or negation)
- ‘$’: ends with
- ‘.’: any single character (except \n)
- ‘*’: match the previous element zero or more times

Example: prepare startup file

Use `sed` to find and replace certain text in file

```
#!/bin/sh

#get script location
script_dir=`cd $(dirname $0) && pwd`
echo "script location is " $script_dir

#get project directory
prot_path=$(echo $script_dir | sed 's|/zzz_.*$||')
echo "project path is" $prot_path
```

1. `cd` to the directory where the script is located.
2. `pwd` print working directory
3. assign the value to variable script_dir
4. remove the last bit of script_dir using `sed`
5. assign the value to variable prot_path

sed: stream editor

sed [option] ‘s/pattern/replacement/’

Example: prepare startup file

```
#find setup file
start_file=`find $prot_path/program -name "startup.*sas"`

if [[ -z $start_file ]]; then
    echo "no setup file found"
else
    echo "setup file is" $start_file
    #update macro variable prot in setup
    sed -i "s|^\(\%let prot\s*\=\).*|\1 $prot_path\;|" $start_file
    #insert new line
    new_line='\%let protpath = \&prot/newfolder\;'
    sed -i "s|^\(\%let prot\s*\=.*\)|\1\n$new_line|" $start_file
    #update libname lptod in setup with &protpath
    sed -i 's|^\(libname outlib\).*|\1 \"\&protpath\"|' $start_file
fi
```

1. find the startup file
2. update &prot with $prot_path
3. add a new line after '%let prot= …'
4. update libname output location to &protpath

```
[wupeik@lctcvp7147 zzz_pw]$ ./startup_update.sh
script location is  /home/wupeik/self_repo/testpath/mk123/zzz_pw
project path is /home/wupeik/self_repo/testpath/mk123
setup file is /home/wupeik/self_repo/testpath/mk123/program/startup.sas
```

Example: frequent data transfer and monitoring task

Monitoring task: comparing current results with last available ones, e.g., last month.

Workflow:

1. Rename old dataset from 2 months ago.
2. Archive last month's dataset to another folder.
3. Copy the latest SDTM data into study folder.
4. Run programs.

Want to do:

Automate step 1 to 3.

Folder A	Folder B
• Dataset from last month • Archived datasets	• Current month's dataset

Example: frequent data transfer and monitoring task

```
[wupeik@lctcvp7266 adamdata]$ ls -l ./archive
total 0
-rw------- 1 wupeik wupeik 0 Aug 18 15:25 data.sas7bdat
```

Want to rename data.sas7bdat in archive folder to

MON_data.sas7bdat

```
[wupeik@lctcvp7147 adamdata]$ ls -l ./archive | awk '{print $3, $6, $9}'

wupeik Aug data.sas7bdat
```

Example: frequent data transfer and monitoring task

```
#!/bin/sh

#get script location
script_dir=`cd $(dirname $0) && pwd`
echo "script location is " $script_dir

#get project directory
prot_path=$(echo $script_dir | sed -e 's|/zzz_.*$||')
echo "project path is" $prot_path

#find data location
data_path=$prot_path/adamdata

if [[ ! -d $data_path ]]; then
    echo "Data folder does not exist"
    exit 1;
fi
```

Exit status

Check exit status from last command

Get month using awk and transform to upper case

```
#get month of the data to be archived
arv_path=$data_path/archive
month=`ls -l $arv_path | grep "data.sas7bdat" | awk '{ print $6 }' | tr [a-z] [A-Z]`
echo "The data to be archived was run in $month"

#change name
mv $arv_path/data.sas7bdat $arv_path/${month}_data.sas7bdat

if [[ $? -ne 0 ]]; then
    echo "Rename failed"
else
    echo "File renamed to ${month}_data.sas7bdat"
fi
```

Example: frequent data transfer and monitoring task

```
#get script location
script_dir=`cd $(dirname $0) && pwd`
echo "script location is " $script_dir

#get project directory
prot_path=$(echo $script_dir | sed -e 's|/zzz_pw$||')
sdtm_path="$prot_path/datasdtm"
echo "sdtm path is " $sdtm_path

#get sdtm input location
sdtm_in="$prot_path-sdtm/datasdtm"
echo "sdtm input from " $sdtm_in

#list sas datasets in sdtm input
echo "sdtm datasets to be copied: "
ls $sdtm_in  | grep \.sas7bdat

#copy files to sdtm folder
cp $(find $sdtm_in -name "*.sas7bdat") $sdtm_path

if [[ $? -ne 0 ]]; then
    echo "copy failed"
else
    echo "copy successful"
fi
```

Fetch SDTM datasets using shell scripts

Advantage:

- Faster
- Less error-prone

Example: frequent data transfer and monitoring task

```
[wupeik@lctcvp7266 adamdata]$ ../zzz_pw/archive_data.sh
script location is  /home/wupeik/self_repo/testpath/mk123/zzz_pw
project path is /home/wupeik/self_repo/testpath/mk123
The data to be archived was run in AUG
File renamed to AUG_data.sas7bdat
[wupeik@lctcvp7266 adamdata]$ ls ./archive
AUG_data.sas7bdat
[wupeik@lctcvp7266 adamdata]$ ../zzz_pw/sdtm_update.sh
script location is  /home/wupeik/self_repo/testpath/mk123/zzz_pw
sdtm path is  /home/wupeik/self_repo/testpath/mk123/datasdtm
sdtm input from  /home/wupeik/self_repo/testpath/mk123-sdtm/datasdtm
sdtm datasets to be copied:
dm.sas7bdat
copy successful
```

Chaining scripts

Manage scripts

```
# archive_data
# archive data in dataanalysis/archive folder
# $1 is path to data folder
# $2 is the name of dataset

archive_data()
{

  if [[ ! -d $1 ]]; then
       echo "Data folder does not exist"
       exit 1;
  fi

  # get month of the data to be archived
  arv_path=$1
  month=`ls -l $arv_path | grep "$2.sas7bdat" | awk '{ print $6 }' | tr [a-z] [A-Z]`
  echo "The data to be archived was run in $month"

  # change name
  mv $arv_path/$2.sas7bdat $arv_path/${month}_${2}.sas7bdat

  if [[ $? -ne 0 ]]; then
       echo "Rename failed"
  else
       echo "File renamed to ${month}_${2}.sas7bdat"
  fi

}
```

```
#!/bin/sh

. ./common.lib

# path defined are
#   script_dir: script location
#   prot_path: protocol folder
#   data_path: ADaM data folder
#   sdtm_path: SDTM data folder
#   sdtm_in: incoming SDTM folder

# archive data
archive_data $data_path data
```

Calling script

Source functions

Integrated workflow

Calling scripts from SAS:

The ‘x’ statement

```
%let dir_path = path_to_script;
x "&dir_path./call_lib.sh";
%include "analysis_program.sas";
```

Integrated workflow

Calling scripts from R:

The system( ) function

```
system (paste0("mkdir -p ", loc, "dir_to_create"))
command <- paste0("command1", var, "command2")
system (command)
```

Summary

1. Shell commands and shell scripting are efficient ways to interface with computers

2. Shell scripting can be useful in automating some repetitive tasks

3. There are multiple ways to run shell scripts

4. Need to take some time to learn writing shell scripts

MERCK
INVENTING FOR LIFE

Thank you