Computational Physics

Prof. Paul Eugenio
Department of Physics
Florida State University
January 10, 2019

http://hadron.physics.fsu.edu/~eugenio/comphy/

Announcements

This week's exercise

- Read Chapter 1
    - Introduction to Computational Physics
        - Turn-In Questions Ch 1 and Unix Intro handout
            - Write down two questions on the material in the chapter and turn it in on Tuesday Jan 15.

- Exercise #0, Unix Navigation
    - see handout

Off-Campus Access

Access the HPC from an off-campus Internet connection by using the FSU VPN Service.

- Go to: https://rcc.fsu.edu/doc/off-campus-vpn-access

A VPN is a virtual private network. Once you install the Cisco VPN application, you just run the program and login to your FSU account. Once done, you will have access to **ssh** into **hpc-login.rcc.fsu.edu**

Mac OSX & Linux users at home (or campus wifi)

- connect to vpn server https://vpn.fsu.edu/hpc
- open a terminal and connect to the hpc using ssh as in class

MS Windows users

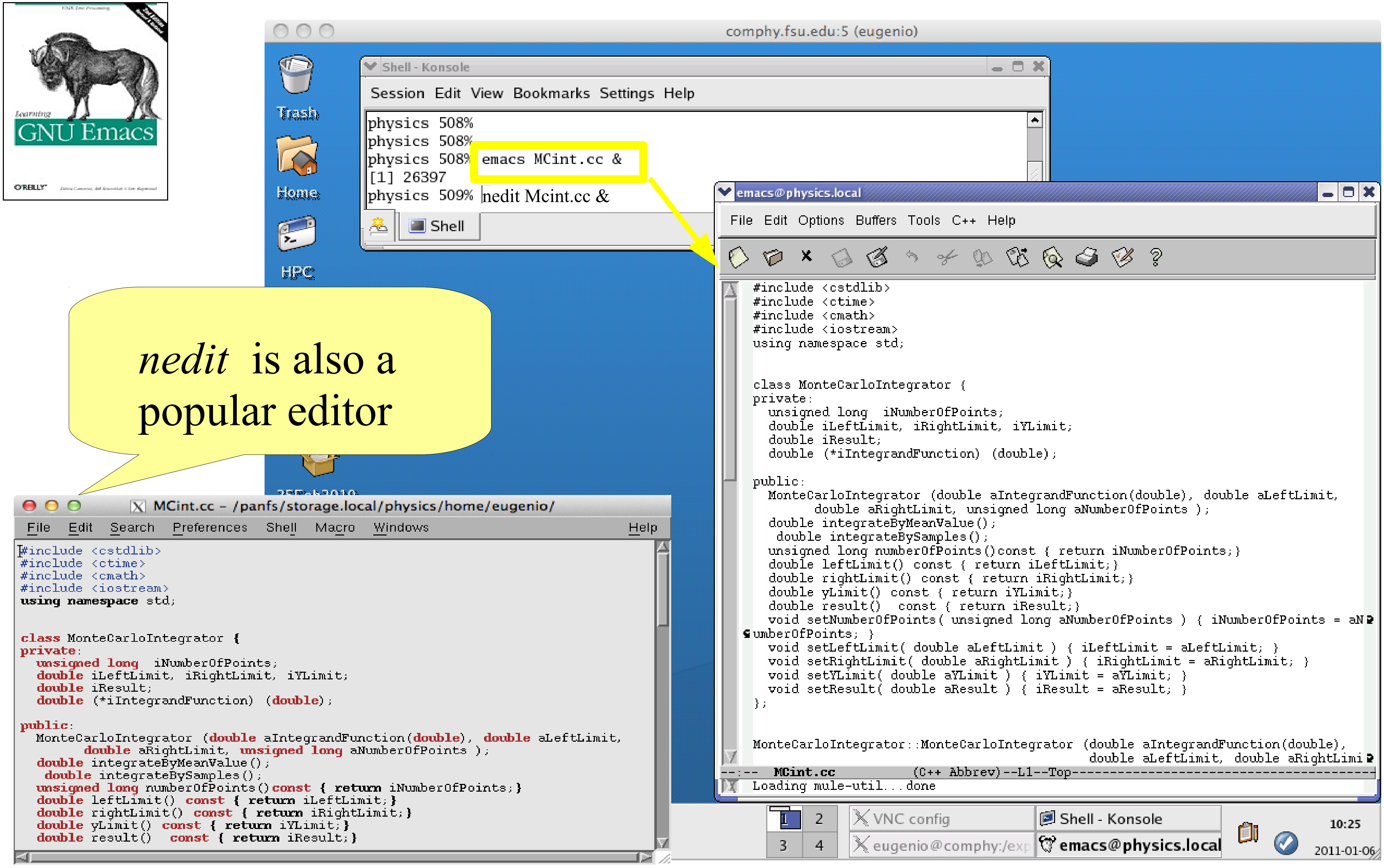
- look into installing ***Cygwin/X***
  - See ***http://x.cygwin.com***
  - Connect to vpn server https://vpn.fsu.edu/hpc before using CygwinX

OK, We should all be up and running.

Unix Basics

Text Editing

Emacs One of the most widely used editors



UNIX Text Processing
Learning
GNU Emacs
O'REILLY
comphy.fsu.edu:5 (eugenio)
Trash
Home
HPC
Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell
emacs@physics.local
File Edit Options Buffers Tools C++ Help
MCint.cc – /panfs/storage.local/physics/home/eugenio/
File Edit Search Preferences Shell Macro Windows Help
VNC config
Shell - Konsole
eugenio@comphy:/exp
emacs@physics.local
1 2 3 4
10:25
2011-01-06

```
physics 508%
physics 508%
physics 508% emacs MCint.cc &
[1] 26397
physics 509% nedit Mcint.cc &
```

nedit is also a popular editor

```
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <iostream>
using namespace std;


class MonteCarloIntegrator {
private:
  unsigned long  iNumberOfPoints;
  double iLeftLimit, iRightLimit, iYLimit;
  double iResult;
  double (*iIntegrandFunction) (double);

public:
  MonteCarloIntegrator (double aIntegrandFunction(double), double aLeftLimit,
        double aRightLimit, unsigned long aNumberOfPoints );
  double integrateByMeanValue();
   double integrateBySamples();
  unsigned long numberOfPoints()const { return iNumberOfPoints;}
  double leftLimit() const { return iLeftLimit;}
  double rightLimit() const { return iRightLimit;}
  double yLimit() const { return iYLimit;}
  double result()  const { return iResult;}
  void setNumberOfPoints( unsigned long aNumberOfPoints ) { iNumberOfPoints = aN
umberOfPoints; }
  void setLeftLimit( double aLeftLimit ) { iLeftLimit = aLeftLimit; }
  void setRightLimit( double aRightLimit ) { iRightLimit = aRightLimit; }
  void setYLimit( double aYLimit ) { iYLimit = aYLimit; }
  void setResult( double aResult ) { iResult = aResult; }
};


MonteCarloIntegrator::MonteCarloIntegrator (double aIntegrandFunction(double),
                                             double aLeftLimit, double aRightLimi
--:--  MCint.cc        (C++ Abbrev)--L1--Top-------------------------------------
Loading mule-util...done
```

```
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <iostream>
using namespace std;


class MonteCarloIntegrator {
private:
  unsigned long  iNumberOfPoints;
  double iLeftLimit, iRightLimit, iYLimit;
  double iResult;
  double (*iIntegrandFunction) (double);

public:
  MonteCarloIntegrator (double aIntegrandFunction(double), double aLeftLimit,
        double aRightLimit, unsigned long aNumberOfPoints );
  double integrateByMeanValue();
   double integrateBySamples();
  unsigned long numberOfPoints()const { return iNumberOfPoints;}
  double leftLimit() const { return iLeftLimit;}
  double rightLimit() const { return iRightLimit;}
  double yLimit() const { return iYLimit;}
  double result()  const { return iResult;}
```

Common File System Navigation

Files

Command		Action
cp	<file1> <file2>	copy <file1> to <file2>
	<files> <dir/>	copy <files> to <dir/>
mv	<file> <dir/>	move <file> to <dir/>
	<file1> <file2>	rename <file1> to <file2>
ls	<files>[or dir/]	list files
ls -l	<dir/>	list files with property info
ls -a	<files>[ordir/]	list invisible content*

* file or directory names starting with a "."

Common File System Navigation

Directories

Command	Action
cd <dir/>	change directory
pwd	print working/current directory
mkdir <name>	make new directory
rmdir <dir/>	remove empty directory
ls -dl <dir/>	list directory properties

Unix Wild Cards

Symbol	Action
?	match any single character
*	match any size string
[abc]	match any enclosed character
[a-f]	match any character in range
[!abc]	match all but enclosed characters
~	current user home directory
~user	home directory of a user

Examples:

```
ls *.py                  # list all files in current directory ending in ".py"
ls  vector[1-5].py       # list files vector1.py, vector2.py, … vector5.py if they exist
ls *.??                  # all files with a two character suffix
```

Unix File Redirection

Symbol	Redirection
>	redirect to standard output
>&	redirect to standard error
>>	append to standard output
|	pipe standard output to another command
|&	pipe standard error to another command
<	input redirection
<<String	read from standard input until "String" is encountered as the only thing on the line.

Examples of file redirection are:

```
cat file1  file2 > file3
cat file1 > file3
cat file2 >> file3
cat file1  file2 | wc -l
wc -l < file3       # same as wc -l file3
```

```
ps aux | grep vnc | grep eugenio
```

Special Symbols

Symbol	Action
;	command separator
&	run command in background **********
&&	run next command upon success
||	run next command if unsuccessful
`command`	execute command first & substitute result
\	escape the following character

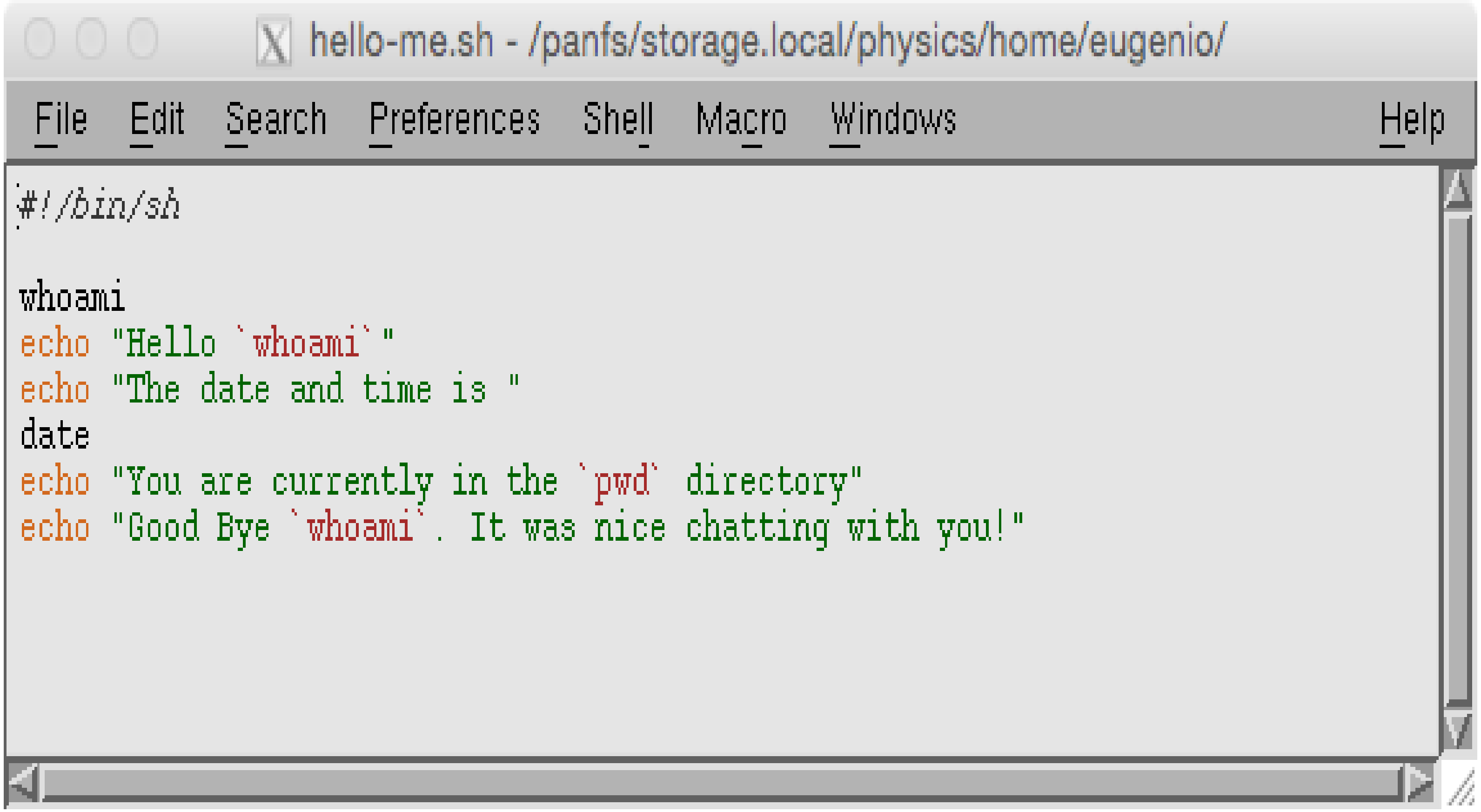
Examples are:

```
cat file1 > file3 ; cat file2 >> file3
emacs  file1 &
grep string file || echo “string not found”
```

Shell Scripts

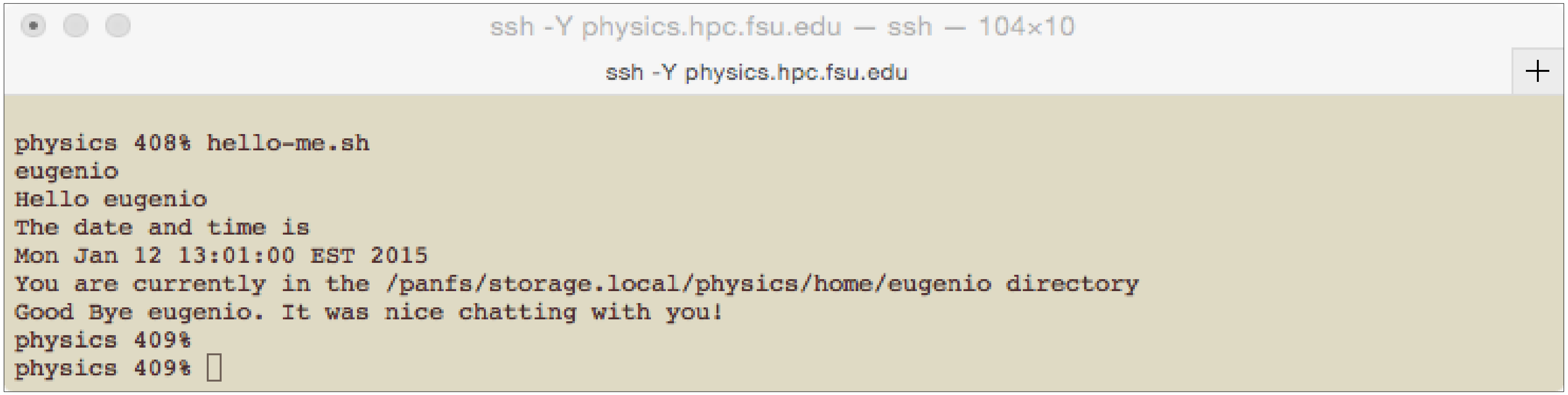
- A shell scripts is a special text file containing a list of shell commands. By executing the script, the shell executes all shell commands line by line.
  - Use an editor like *emacs* or *nedit* to write a shell script
    - The first line of the shell script should be "***#! /bin/sh***"
  - After writing shell script set the execute permission for your script
    - syntax: *chmod +x your-script-name.sh*
  - Execute your script
    - syntax: *./your-script-name.sh*

Shell Script Example



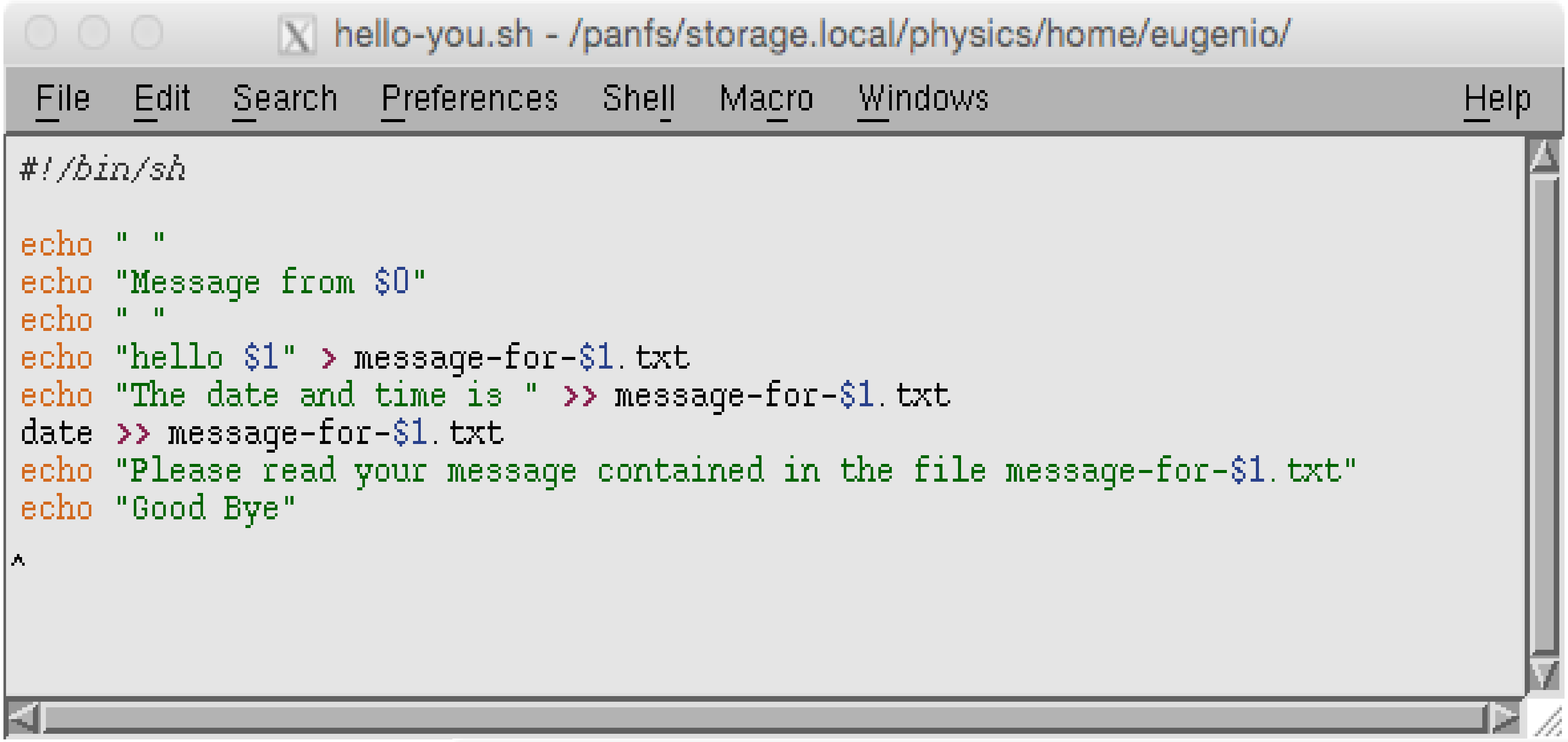
```
#!/bin/sh

whoami
echo "Hello `whoami`"
echo "The date and time is "
date
echo "You are currently in the `pwd` directory"
echo "Good Bye `whoami`. It was nice chatting with you!"
```


ssh -Y physics.hpc.fsu.edu — ssh — 104×10
ssh -Y physics.hpc.fsu.edu

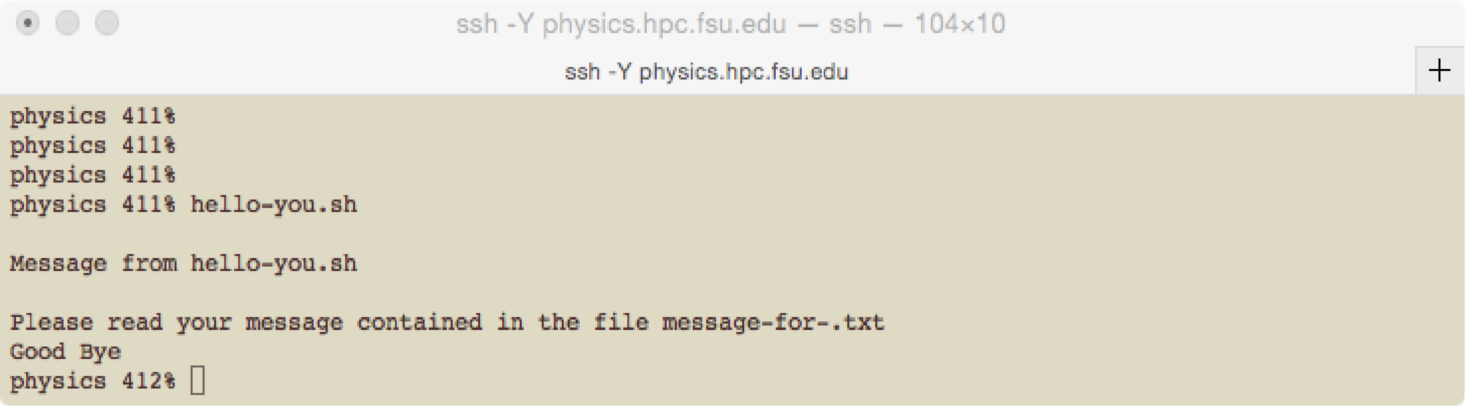
```
physics 408% hello-me.sh
eugenio
Hello eugenio
The date and time is
Mon Jan 12 13:01:00 EST 2015
You are currently in the /panfs/storage.local/physics/home/eugenio directory
Good Bye eugenio. It was nice chatting with you!
physics 409%
physics 409% 
```

Shell Script with $0, $1, $2, … Command Line Variables



```
#!/bin/sh

echo " "
echo "Message from $0"
echo " "
echo "hello $1" > message-for-$1.txt
echo "The date and time is " >> message-for-$1.txt
date >> message-for-$1.txt
echo "Please read your message contained in the file message-for-$1.txt"
echo "Good Bye"
```


ssh -Y physics.hpc.fsu.edu — ssh — 104×10
ssh -Y physics.hpc.fsu.edu
+

```
physics 411%
physics 411%
physics 411%
physics 411% hello-you.sh

Message from hello-you.sh

Please read your message contained in the file message-for-.txt
Good Bye
physics 412%
```

Today's Exercise

- Exercise #0

Unix Navigation