

# UNIX Shell-Scripting

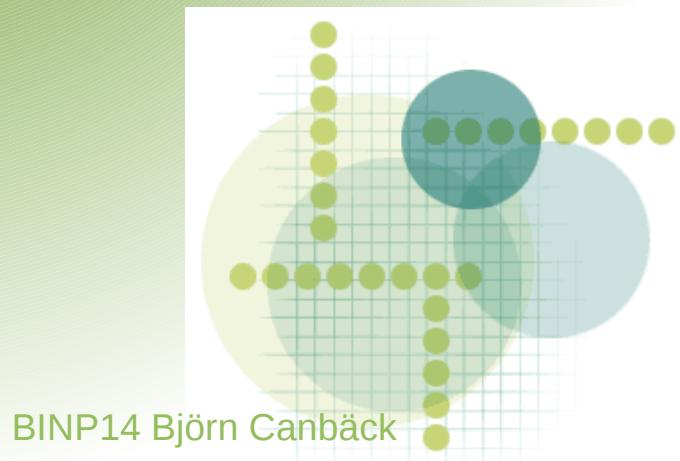
With focus on bash

BINP14 Björn Canbäck



# Outline

- What is a shell? A shell script?
- Introduction to bash
- Running Commands



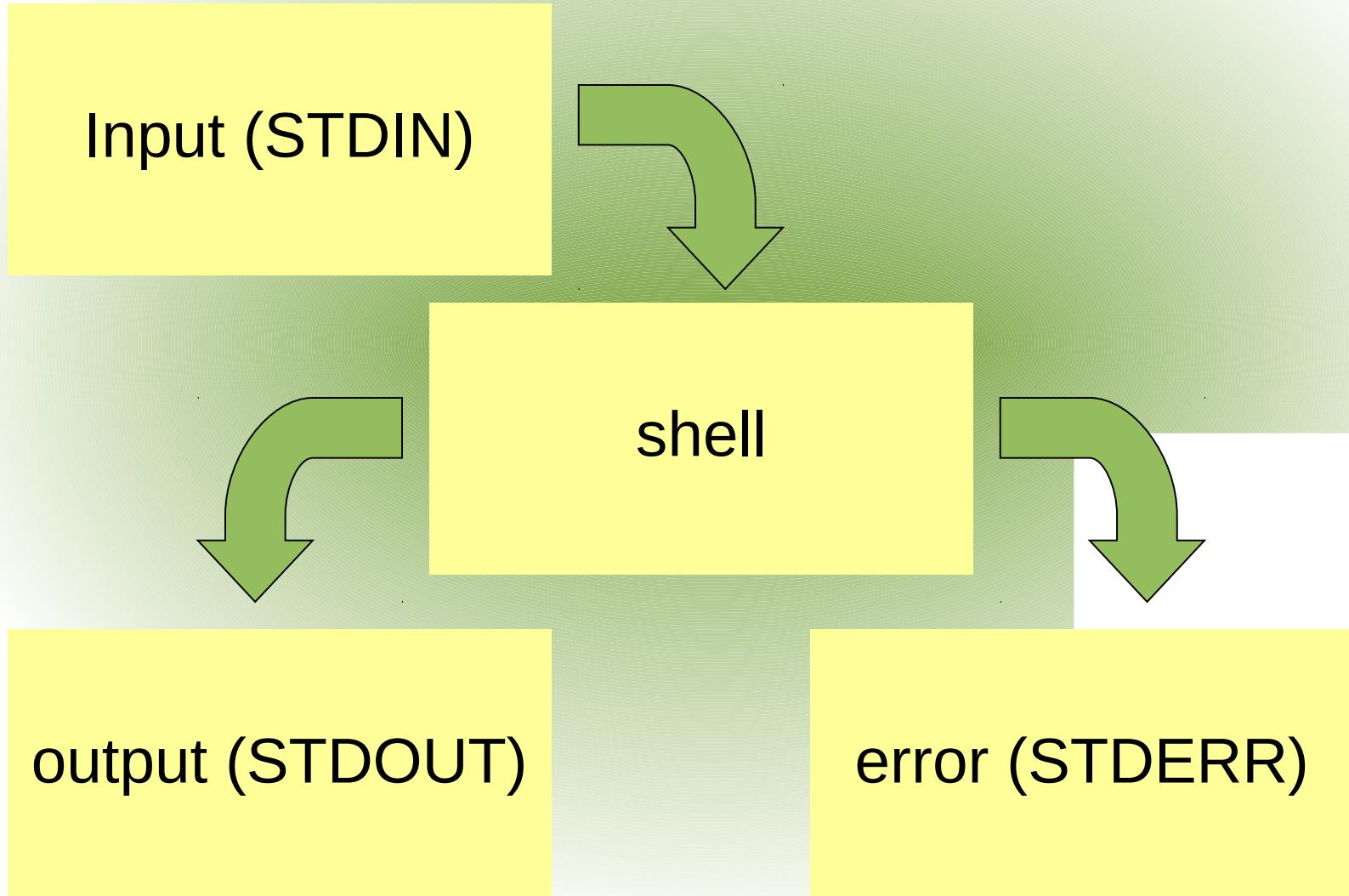
# What is a shell?

A Unix shell is a command-line interpreter or shell that provides a traditional user interface for the Unix operating system and for Unix-like systems. Users direct the operation of the computer by entering commands as text for a command line interpreter to execute or by creating text scripts of one or more such commands.

Source: [http://en.wikipedia.org/wiki/Unix\\_shell](http://en.wikipedia.org/wiki/Unix_shell)



# What is a shell?



# Common Shells

- Bash (/bin/bash) Bourne again shell
- C Shell (/bin/csh)
- Turbo C Shell (/bin/tcsh)
- Korn Shell (/bin/ksh)



BINP14 Björn Canbäck

# What is bin ?

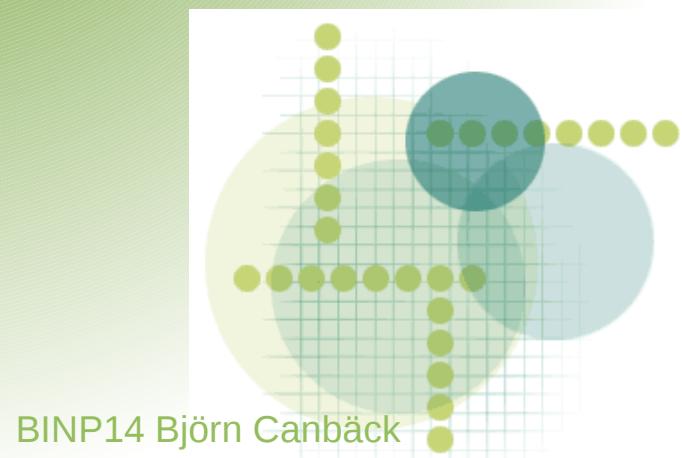
- /bin
- /usr/bin
- /usr/local/bin
- /home/bjorn/bin



BINP14 Björn Canbäck

# What is a shell script?

- A text file
- With instructions
- Executable



# What is a Shell Script?

```
% cat > hello.sh <<HERE
#!/bin/sh
echo 'Hello world!'
HERE
% chmod +x hello.sh
% ./hello.sh
Hello world!
```



# What is a Shell Script? A Text File

```
% cat > hello.sh <<HERE
```

```
#!/bin/sh
```

```
echo 'Hello world!'
```

```
HERE
```

```
% chmod +x hello.sh
```

```
% ./hello.sh
```

```
Hello world!
```



# What is a Shell Script? How To Run

```
% cat > hello.sh <<HERE  
#!/bin/sh  
echo 'Hello world!'  
HERE  
  
% chmod +x hello.sh  
  
% ./hello.sh  
Hello world!
```



BINP14 Björn Canbäck

# What is a Shell Script? What To Do

```
% cat > hello.sh <<HERE  
#!/bin/sh  
echo 'Hello world!'  
HERE  
% chmod +x hello.sh  
% ./hello.sh  
Hello world!
```



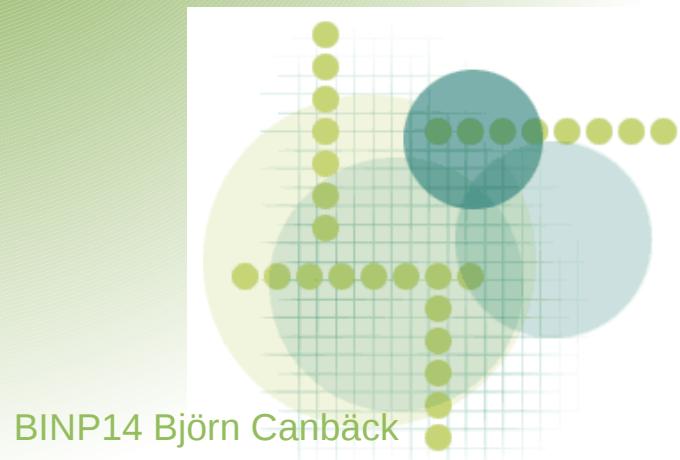
# What is a Shell Script? Executable

```
% cat > hello.sh <<HERE
#!/bin/sh
echo 'Hello world!'
HERE
% chmod +x hello.sh
% ./hello.sh
Hello world!
```



# What is a Shell Script? Running it

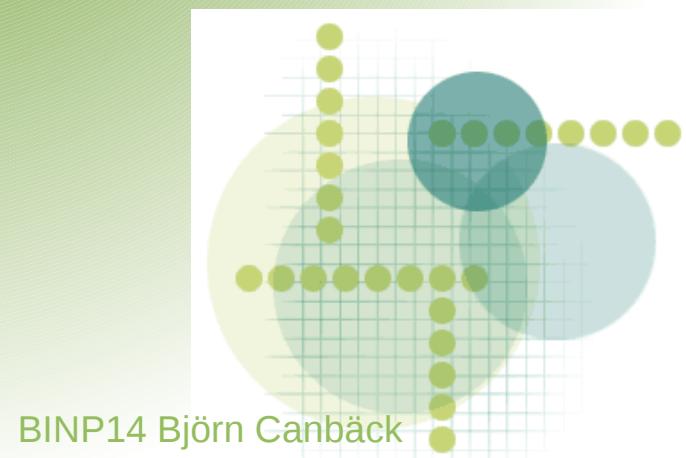
```
% cat > hello.sh <<HERE  
#!/bin/sh  
echo 'Hello world'  
HERE  
% chmod +x hello.sh  
% ./hello.sh  
Hello world!
```



BINP14 Björn Canbäck

# Finding the program: PATH

- % ./hello.sh
- % echo \$PATH  
/bin:/usr/bin:/usr/local/bin:  
/home/bjorn/bin
- % which echo  
/usr/bin/echo



# Variables and the environment

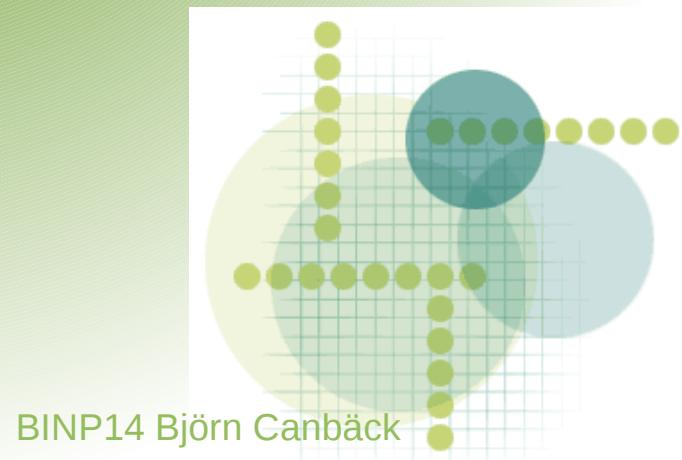
```
% hello.sh
```

```
bash: hello.sh: Command not found
```

```
% PATH="$PATH:."
```

```
% hello.sh
```

```
Hello, world
```



# Redirection

```
echo hej > test.txt
```

```
echo " hej" >> test.txt
```

Expert users only:

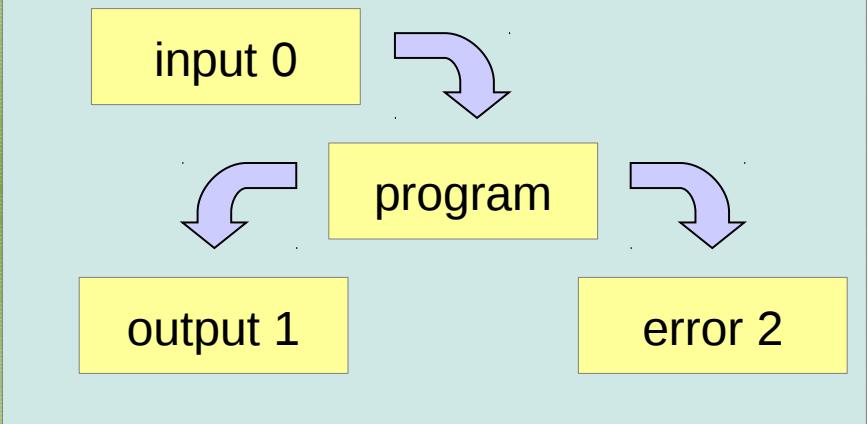
```
cat < test.txt
```

```
cat <<INPUT  
Some input  
INPUT
```

```
test.sh 2> myError
```

```
text.sh> myErrorAndOut 2>&1
```

Expert users only:



# Quoting

```
% echo '$USER'  
$USER  
% echo "$USER"  
bjorn  
% echo $USER  
bjorn  
% echo \"  
"  
% echo \>  
>
```



BINP14 Björn Canbäck

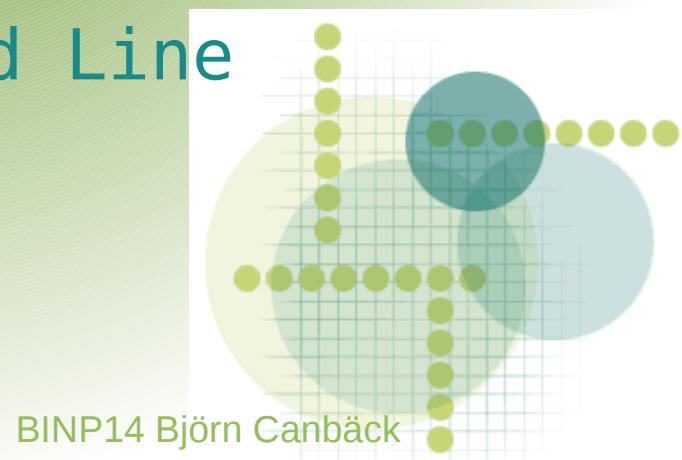
# How to learn

- man
  - man bash
  - man cat
  - man man
- *Learning the Bash Shell*, 2<sup>nd</sup> Ed.
- “Bash Reference” Cards



## Continuing lines: \

```
% echo This \
Is \
A \
Very \
Long \
Command Line
This Is A Very Long Command Line
%
```



# Make Your Life Easier

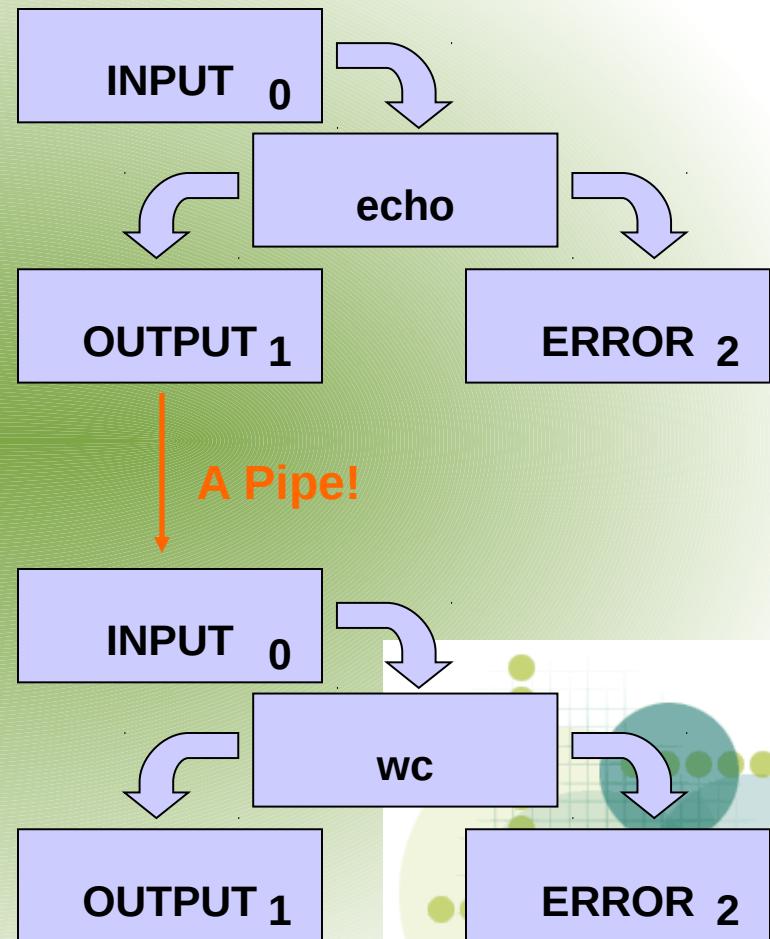
- TAB completion
- Control+R
- Control+S



# Pipes

- Lots of Little Tools

```
echo "Hello" | \  
WC - C
```

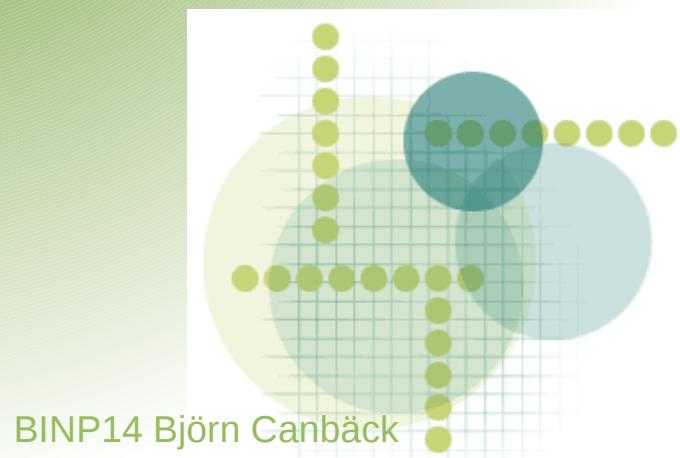


Following is only if you want to learn more

# Exit status (expert users)

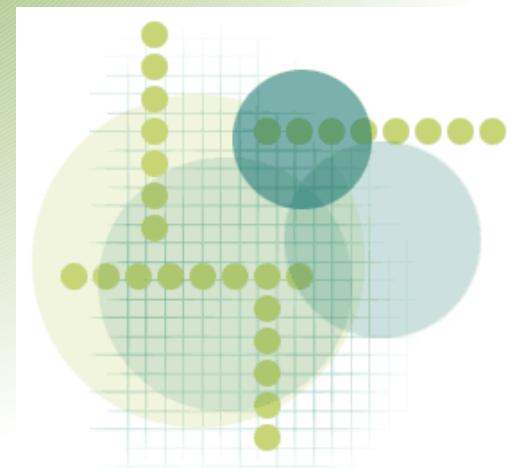
- \$?
- 0 is True

```
% ls /does/not/exist  
% echo $?  
1  
% echo $?  
0
```



## Exit status: (expert users)

```
% cat > test.sh <<_TEST_
exit 3
_TEST_
% chmod +x test.sh
% ./test.sh
% echo $?
3
```



## Logic: test (expert users)

```
% test 1 -lt 10  
% echo $?  
0  
% test 1 == 10  
% echo $?  
1
```



# Logic: test (expert users)

- `test`
- `[ ]`
  - `[ 1 -lt 10 ]`
- `[[ ]]`
  - `[[ "this string" =~ "this" ]]`
- `(( ))`
  - `(( 1 < 10 ))`



## Logic: test (expert users)

- [ -f /etc/passwd ]
- [ ! -f /etc/passwd ]
- [ -f /etc/passwd -a -f /etc/shadow ]
- [ -f /etc/passwd -o -f /etc/shadow ]



## An aside: \$(()) for Math (expert users)

```
% echo $(( 1 + 2 ))  
3  
% echo $(( 2 * 3 ))  
6  
% echo $(( 1 / 3 ))  
0
```



## Logic: if (expert users)

```
if something
then
:
# “elif” a contraction of “else if”:
elif something-else
then
:
else
then
:
fi
```



## Logic: if (expert users)

```
if [ $USER -eq "borwicjh" ]
then
:
# "elif" a contraction of "else if":
elif ls /etc/oratab
then
:
else
then
:
fi
```



## Logic: if (expert users)

```
# see if a file exists  
if [ -e /etc/passwd ]  
then  
    echo "/etc/passwd exists"  
else  
    echo "/etc/passwd not found!"  
fi
```



## Logic: for (expert users)

```
for i in 1 2 3  
do  
    echo $i  
done
```



## Logic: for (expert users)

```
for i in /*  
do  
    echo "Listing $i:"  
    ls -l $i  
    read  
done
```



## Logic: for (expert users)

```
for i in /*  
do  
    echo "Listing $i:"  
    ls -l $i  
    read  
done
```



## Logic: for (expert users)

```
for i in /*  
do  
    echo "Listing $i:"  
    ls -l $i  
    read  
done
```



## Logic: C-style for (expert users)

```
for (( expr1      ;  
      expr2      ;  
      expr3      ))  
  
do  
    list  
done
```



## Logic: C-style for (expert users)

```
LIMIT=10  
for (( a=1 ;  
      a<=LIMIT ;  
      a++ ))  
  
do  
    echo -n "$a "  
done
```



## Logic: while

`while something`

`do`

`:`

`done`



## Logic: while

```
a=0; LIMIT=10
while [ "$a" -lt "$LIMIT" ]
do
    echo -n "$a "
    a=$(( a + 1 ))
done
```



# Counters

```
COUNTER=0  
while [ -e "$FILE.COUNTER" ]  
do  
    COUNTER=$(( COUNTER + 1 ))  
done
```

- Note: race condition



# Reusing Code: “Sourcing”

```
% cat > /path/to/my/passwords << PW_  
FTP_USER="sct"  
PW_  
% echo $FTP_USER  
  
% . /path/to/my/passwords  
% echo $FTP_USER  
sct  
%
```

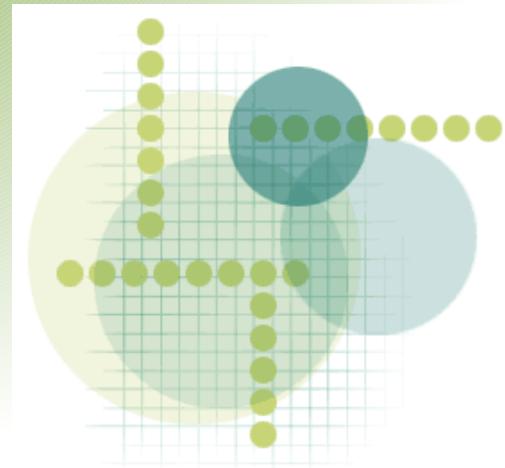


# Variable Manipulation

```
% FILEPATH=/path/to/my/output.lis
% echo $FILEPATH
/path/to/my/output.lis
% echo ${FILEPATH%.lis}
/path/to/my/output
% echo ${FILEPATH#*/}
path/to/my/output.lis
% echo ${FILEPATH##*/}
output.lis
```



# Running Programs



# Reasons for Running Programs

- Check Return Code
  - `$?`
- Get Job Output
  - `OUTPUT=`echo "Hello"``
  - `OUTPUT=$(echo "Hello")`
- Send Output Somewhere
  - Redirection: `<`, `>`
  - Pipes



# Email Notification

```
% echo "Message" | \
mail -s "Here's your message" \
borwicjh@wfu.edu
```



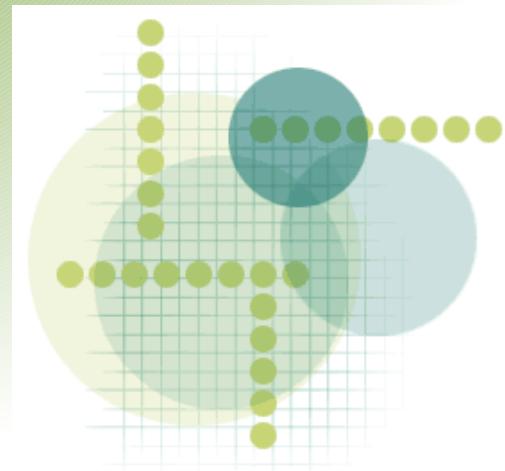
# Dates

```
% DATESTRING=`date +%Y%m%d`  
% echo $DATESTRING  
20060125  
% man date
```



# FTP the Hard Way

```
ftp -n -u server.wfu.edu <<_FTP_
user username password
put FILE
_FTP_
```



# FTP with wget

- `wget \`  
`ftp://user:pass@server.wfu.edu/file`
- `wget -r \`  
`ftp://user:pass@server.wfu.edu/dir/`



## FTP with curl

```
curl -T upload-file \  
-u username:password \  
ftp://server.wfu.edu/dir/file
```



## Searching: find

```
% find /home/borwicjh \  
    -name '*.lis'
```

*[all files matching \*.lis]*

```
% find /home/borwicjh \  
    -mtime -1 -name '*.lis'
```

*[\*.lis, if modified within 24h]*

```
% man find
```

