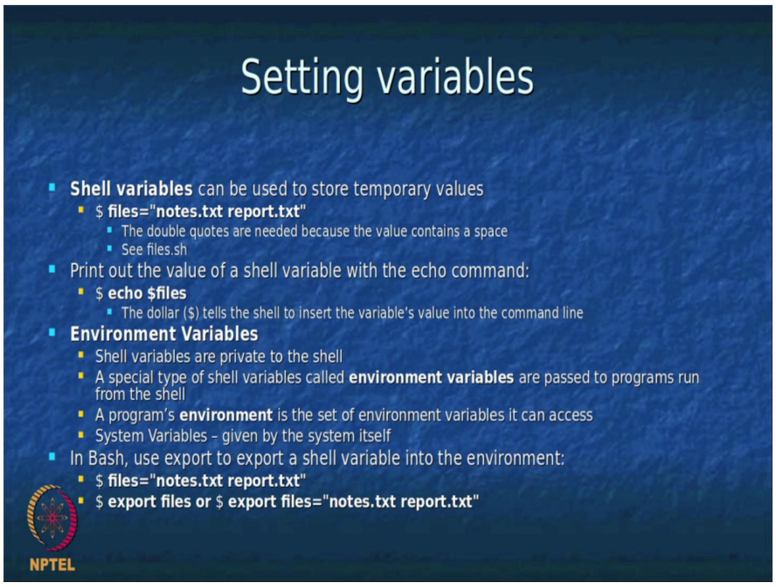
Information Security
Sri M J Shankar Raman,
Consultant of Computer Science and Engineering
Indian Institute of Technology Madras
Module 29
Variables (Continued)

Ok in the last session we had a brief introduction about the variables how to read a variable how to print a variable. Now we will expand this topic of variables.

(Refer Slide Time: 00:22)

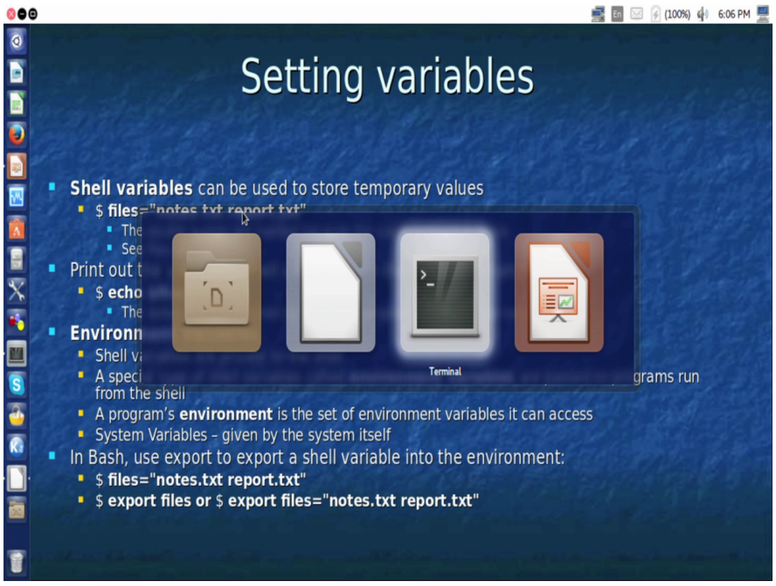


Setting variables

- **Shell variables** can be used to store temporary values
  - $ **files="notes.txt report.txt"**
    - The double quotes are needed because the value contains a space
    - See files.sh
- Print out the value of a shell variable with the echo command:
  - $ **echo $files**
    - The dollar ($) tells the shell to insert the variable's value into the command line
- **Environment Variables**
  - Shell variables are private to the shell
  - A special type of shell variables called **environment variables** are passed to programs run from the shell
  - A program's **environment** is the set of environment variables it can access
  - System Variables - given by the system itself
- In Bash, use export to export a shell variable into the environment:
  - $ **files="notes.txt report.txt"**
  - $ **export files or** $ **export files="notes.txt report.txt"**

We also found out how to set variables and we will look at slightly different examples. So one of the things that we mentioned is that you can declare any variable in shell at any point in time you need not have to tell what is the data type etc. So here is one example so shell variables can be used to show temporary values and that is what we have been doing in the previous example. So here is a line which says that files is equal to within double quote here notes dot text and report dot text. And if you remember I told you that whenever you assign a variable a value then the whole thing goes as a string. So in this case if you try to print this file it will actually print it as notes dot text and report dot text.

(Refer Slide Time: 01:15)

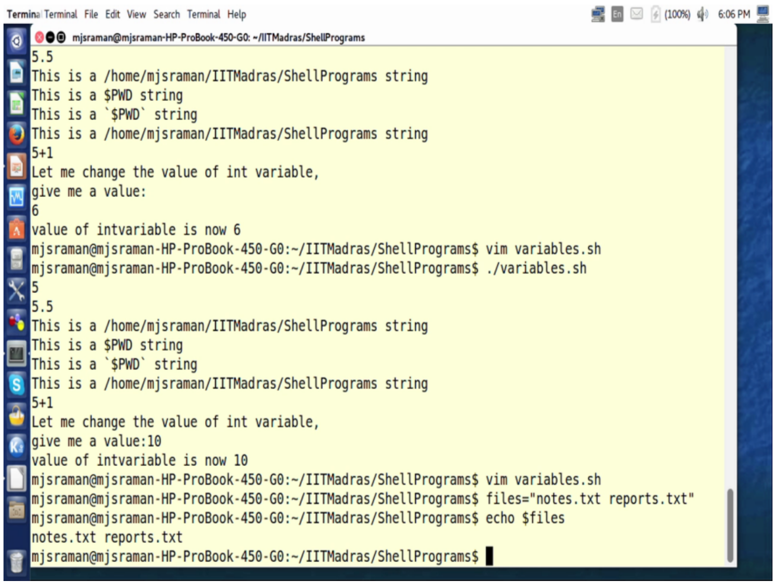


Setting variables

- **Shell variables** can be used to store temporary values
  - $ **files**="notes.txt report.txt"
    - The
    - See
- Print out t
  - $ **echo**
    - The
- **Environm**
  - Shell v
  - A speci grams run from the shell
  - A program's **environment** is the set of environment variables it can access
  - System Variables - given by the system itself
- In Bash, use export to export a shell variable into the environment:
  - $ **files="notes.txt report.txt"**
  - $ **export files or** $ **export files="notes.txt report.txt"**

So next see it as an example here

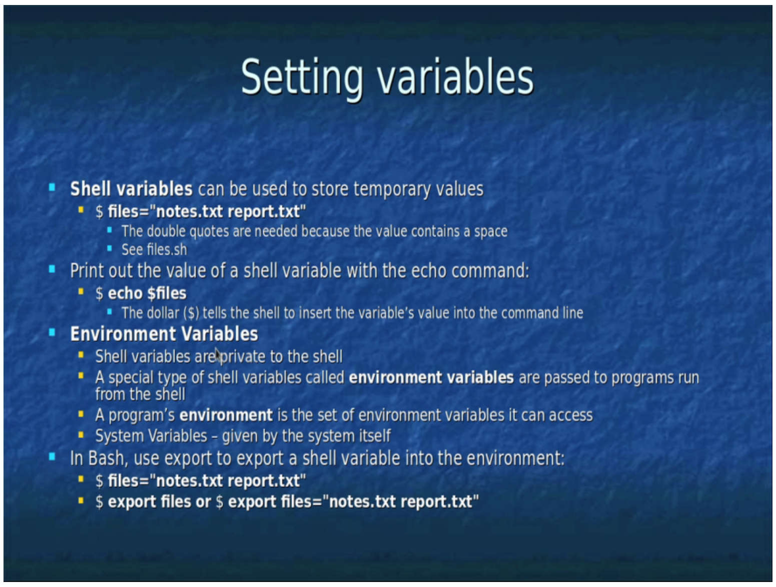
(Refer Slide Time: 01:20)



```
5.5
This is a /home/mjsraman/IITMadras/ShellPrograms string
This is a $PWD string
This is a `$PWD` string
This is a /home/mjsraman/IITMadras/ShellPrograms string
5+1
Let me change the value of int variable,
give me a value:
6
value of intvariable is now 6
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim variables.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ ./variables.sh
5
5.5
This is a /home/mjsraman/IITMadras/ShellPrograms string
This is a $PWD string
This is a `$PWD` string
This is a /home/mjsraman/IITMadras/ShellPrograms string
5+1
Let me change the value of int variable,
give me a value:10
value of intvariable is now 10
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim variables.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ files="notes.txt reports.txt"
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $files
notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ 
```

So let us say files is equal to notes dot text and report dot text, now if I echo this, you see that it just prints whatever we had given so that is what we have been telling that it interpresent as a string.

(Refer Slide Time: 01:40)

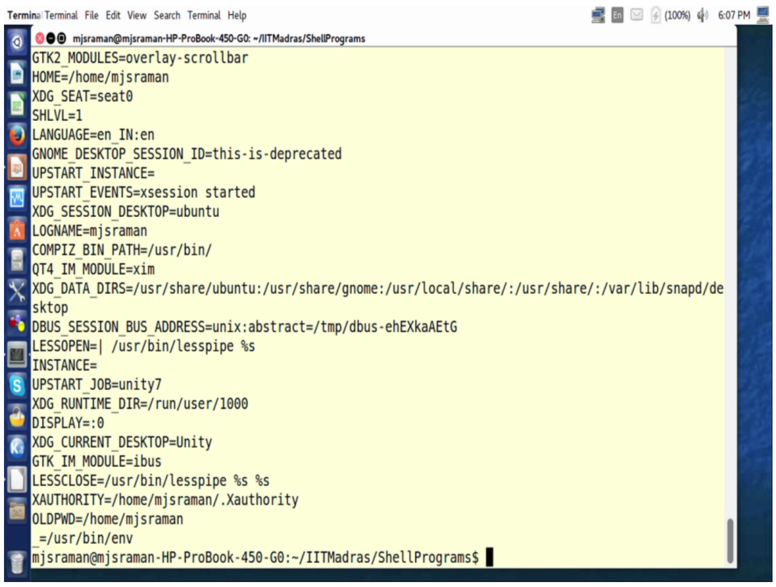


Setting variables

- **Shell variables** can be used to store temporary values
  - $ **files="notes.txt report.txt"**
    - The double quotes are needed because the value contains a space
    - See files.sh
- Print out the value of a shell variable with the echo command:
  - $ **echo $files**
    - The dollar ($) tells the shell to insert the variable's value into the command line
- **Environment Variables**
  - Shell variables are private to the shell
  - A special type of shell variables called **environment variables** are passed to programs run from the shell
  - A program's **environment** is the set of environment variables it can access
  - System Variables - given by the system itself
- In Bash, use export to export a shell variable into the environment:
  - $ **files="notes.txt report.txt"**
  - $ **export files or** $ **export files="notes.txt report.txt"**

Now and we have printed it and we have found out that how it works ok? So one of the things is you have variables which are called environment variables. So one example we saw in the last session was that we had talked about variable called dollar pwd as a present working directory ok? So if the environment variables are something like this say for example in this room I am dissociated from what is happening outside this room so this becomes my environment. So in initial script when you are running your program ok you have run it with a constraint space or environment and this space provides certain variables which you can use one of the variables we saw was suppose the shell wants to know who is running this program ok? So it can echo dollar log name. Similarly the shell wants to know in currently it is in which directory.

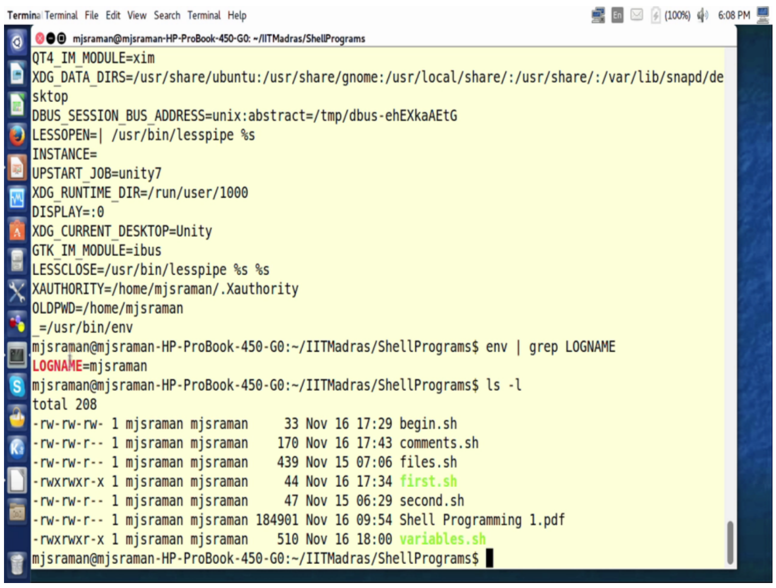
(Refer Slide Time: 02:40)


Terminal Terminal File Edit View Search Terminal Help
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms
(100%) 6:07 PM

```
GTK2_MODULES=overlay-scrollbar
HOME=/home/mjsraman
XDG_SEAT=seat0
SHLVL=1
LANGUAGE=en_IN:en
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
UPSTART_INSTANCE=
UPSTART_EVENTS=xsession started
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=mjsraman
COMPIZ_BIN_PATH=/usr/bin/
QT4_IM_MODULE=xim
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/var/lib/snapd/de
sktop
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-ehEXkaAEtG
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/mjsraman/.Xauthority
OLDPWD=/home/mjsraman
_=/usr/bin/env
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ 
```

So for example let us try suppose the shell now this is the shell and this shell wants to know which directory it is in currently because if you remember there is no interaction that happens when you run the shell. So I can do that I can do echo dollar pwd and it tells you that currently this shell is I am here in this shell in this place ok?. So suppose I want to find out what is my name as last time So it says that my name is mjsraman.

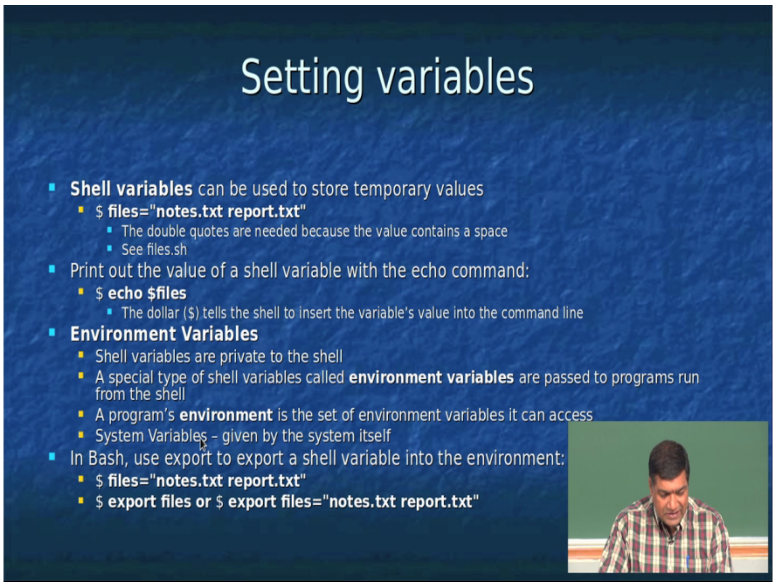
(Refer Slide Time: 03:15)


Terminal Terminal File Edit View Search Terminal Help
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms
(100%) 6:08 PM

```
QT4_IM_MODULE=xim
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/var/lib/snapd/de
sktop
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-ehEXkaAEtG
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/mjsraman/.Xauthority
OLDPWD=/home/mjsraman
_=/usr/bin/env
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep LOGNAME
LOGNAME=mjsraman
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ ls -l
total 208
-rw-rw-rw- 1 mjsraman mjsraman     33 Nov 16 17:29 begin.sh
-rw-rw-r-- 1 mjsraman mjsraman    170 Nov 16 17:43 comments.sh
-rw-rw-r-- 1 mjsraman mjsraman    439 Nov 15 07:06 files.sh
-rwxrwxr-x 1 mjsraman mjsraman     44 Nov 16 17:34 first.sh
-rw-rw-r-- 1 mjsraman mjsraman     47 Nov 15 06:29 second.sh
-rw-rw-r-- 1 mjsraman mjsraman 184901 Nov 16 09:54 Shell Programming 1.pdf
-rwxrwxr-x 1 mjsraman mjsraman    510 Nov 16 18:00 variables.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ 
```

So if I type a command called env it tells you and you can see that in this env look at this this variable pwd this is what we saw then I can also look at environmental command and then grep for log name so if I get for log name it prints mjsraman. So if you look at this these are the environmental variables and there are lot many of them. You can have one session interpreting what this all these things are? But any how we will take somehow those things that are important to us. So here is an example so if you llook at this it says the language is English and then Indian English ok? So such things what is my home directory it gives if you type echo dollar home you will get is so on. So these are the environmental variables that you have in your wishing ok?

(Refer Slide Time: 04:18)

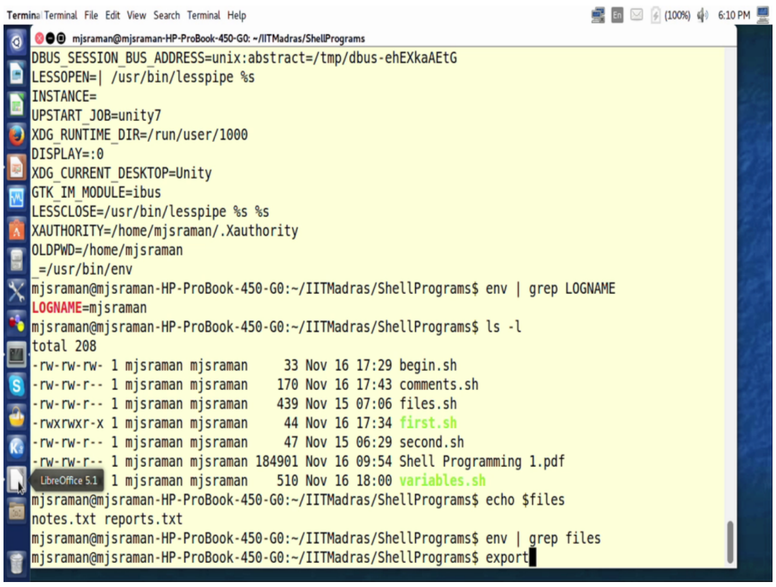


Setting variables

- **Shell variables** can be used to store temporary values
  - $ **files="notes.txt report.txt"**
    - The double quotes are needed because the value contains a space
    - See files.sh
- Print out the value of a shell variable with the echo command:
  - $ **echo $files**
    - The dollar ($) tells the shell to insert the variable's value into the command line
- **Environment Variables**
  - Shell variables are private to the shell
  - A special type of shell variables called **environment variables** are passed to programs run from the shell
  - A program's **environment** is the set of environment variables it can access
  - System Variables – given by the system itself
- In Bash, use export to export a shell variable into the environment:
  - $ **files="notes.txt report.txt"**
  - $ **export files** or $ **export files="notes.txt report.txt"**

Ok now so the shell variables are private to the shell that is if you run two shells then the second shell does not know about the variables that are present in the first shell unless you can export the variable to the environment and the other shell takes that environment, ok? So these environment variables are passed to programs run from the shell, so the reason why So i can even make a private shell variable and then I can export it to the environment so that any program that runs in that shell can take this private variable, ok so let us take a look at this example so if you look at this we have this files is equal to " notes dot text and space report dot text" and when I say export files what it does is this file variable goes and sits as an environmental variable. So let us demonstrate them with an example to understand this.

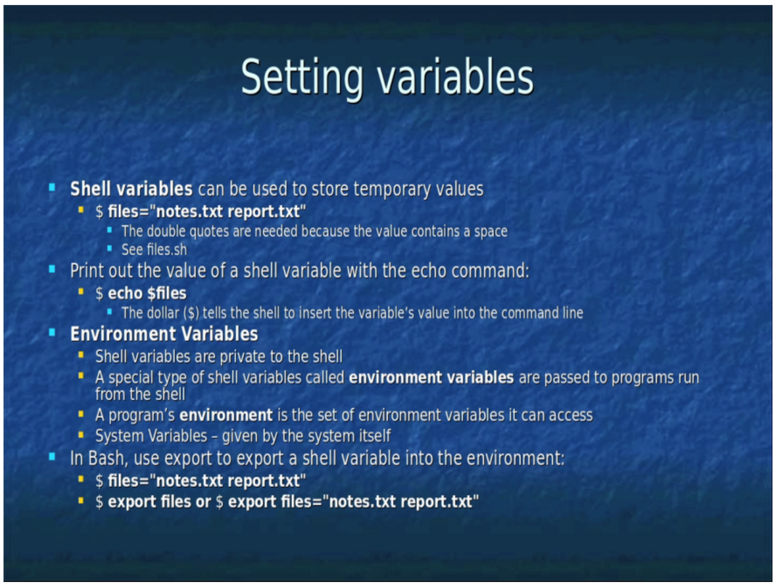
(Refer Slide Time: 05:18)


Terminal Terminal File Edit View Search Terminal Help
(100%) 6:10 PM
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms
LibreOffice 5.1

```
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-ehEXkaAEtG
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/mjsraman/.Xauthority
OLDPWD=/home/mjsraman
_=/usr/bin/env
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep LOGNAME
LOGNAME=mjsraman
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ ls -l
total 208
-rw-rw-rw- 1 mjsraman mjsraman     33 Nov 16 17:29 begin.sh
-rw-rw-r-- 1 mjsraman mjsraman    170 Nov 16 17:43 comments.sh
-rw-rw-r-- 1 mjsraman mjsraman    439 Nov 15 07:06 files.sh
-rwxrwxr-x 1 mjsraman mjsraman     44 Nov 16 17:34 first.sh
-rw-rw-r-- 1 mjsraman mjsraman     47 Nov 15 06:29 second.sh
-rw-rw-r-- 1 mjsraman mjsraman 184901 Nov 16 09:54 Shell Programming 1.pdf
           1 mjsraman mjsraman    510 Nov 16 18:00 variables.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $files
notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ export
```

So if you look at this echo dollar files so the value is here now suppose I put environmental command and then search for files it is not there correct? So now what I will do is I will do an export ok?

(Refer Slide Time: 05:40)

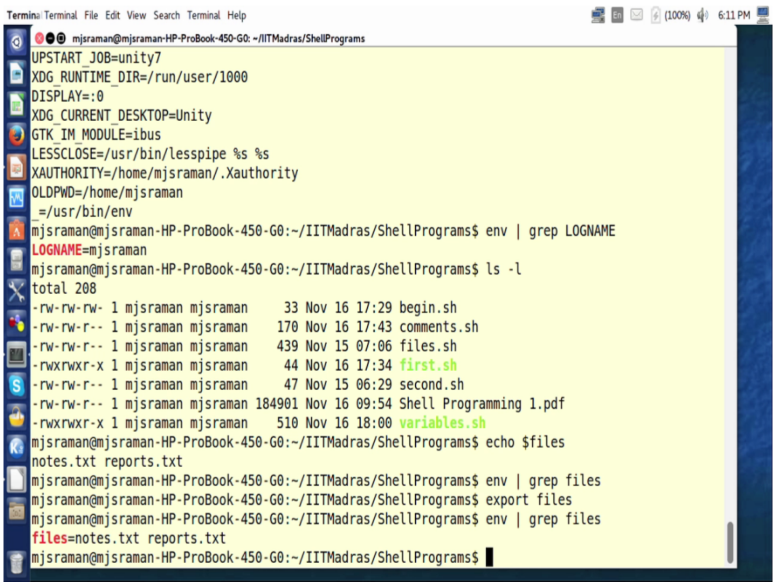


Setting variables

- **Shell variables** can be used to store temporary values
  - $ **files="notes.txt report.txt"**
    - The double quotes are needed because the value contains a space
    - See files.sh
- Print out the value of a shell variable with the echo command:
  - $ **echo $files**
    - The dollar ($) tells the shell to insert the variable's value into the command line
- **Environment Variables**
  - Shell variables are private to the shell
  - A special type of shell variables called **environment variables** are passed to programs run from the shell
  - A program's **environment** is the set of environment variables it can access
  - System Variables - given by the system itself
- In Bash, use export to export a shell variable into the environment:
  - $ **files="notes.txt report.txt"**
  - $ **export files or** $ **export files="notes.txt report.txt"**

So if you look at this command it says export files or export files is equal to there are two syntaxes that are available since we already assign them the value we will use this syntax ok?

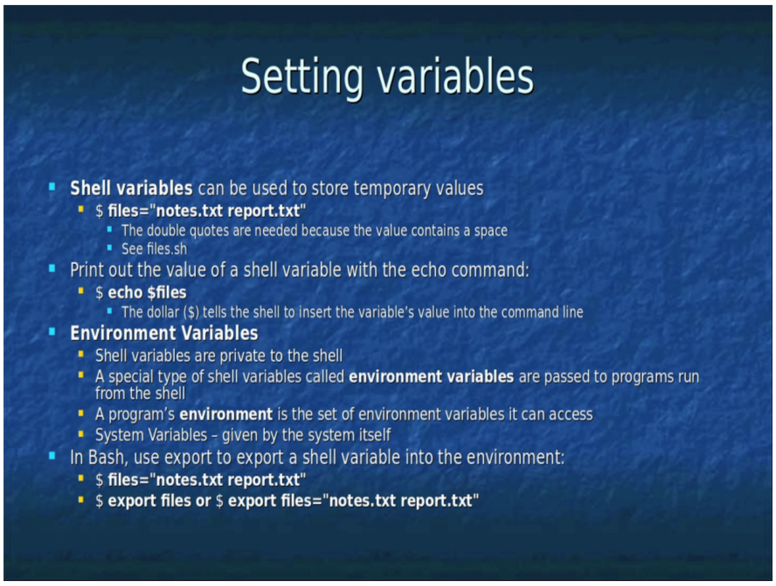
(Refer Slide Time: 05:50)



```
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/mjsraman/.Xauthority
OLDPWD=/home/mjsraman
_=/usr/bin/env
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep LOGNAME
LOGNAME=mjsraman
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ ls -l
total 208
-rw-rw-rw- 1 mjsraman mjsraman     33 Nov 16 17:29 begin.sh
-rw-rw-r-- 1 mjsraman mjsraman    170 Nov 16 17:43 comments.sh
-rw-rw-r-- 1 mjsraman mjsraman    439 Nov 15 07:06 files.sh
-rwxrwxr-x 1 mjsraman mjsraman     44 Nov 16 17:34 first.sh
-rw-rw-r-- 1 mjsraman mjsraman     47 Nov 15 06:29 second.sh
-rw-rw-r-- 1 mjsraman mjsraman 184901 Nov 16 09:54 Shell Programming 1.pdf
-rwxrwxr-x 1 mjsraman mjsraman    510 Nov 16 18:00 variables.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $files
notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ export files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
files=notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$
```

So we will do export files ok now let me run this command so you see that a variable that was local to this shell actually got exported as an environmental variable. So what happens is that if I run a program so this is useful suppose I am running 4 or 5 programs and I want to initialize certain variables and pass it to the new programs this is one way of doing it ok?

(Refer Slide Time: 06:24)

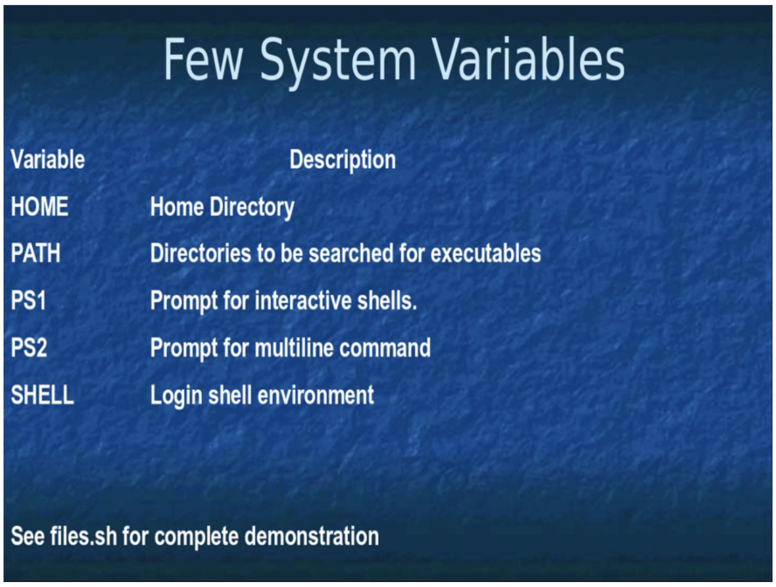


Setting variables

- **Shell variables** can be used to store temporary values
  - $ **files="notes.txt report.txt"**
    - The double quotes are needed because the value contains a space
    - See files.sh
- Print out the value of a shell variable with the echo command:
  - $ **echo $files**
    - The dollar ($) tells the shell to insert the variable's value into the command line
- **Environment Variables**
  - Shell variables are private to the shell
  - A special type of shell variables called **environment variables** are passed to programs run from the shell
  - A program's **environment** is the set of environment variables it can access
  - System Variables - given by the system itself
- In Bash, use export to export a shell variable into the environment:
  - $ **files="notes.txt report.txt"**
  - $ **export files or** $ **export files="notes.txt report.txt"**

So you can set the variables and then export it for other export it to the environment so that other commands can use that variable ok?

(Refer Slide Time: 06:32)



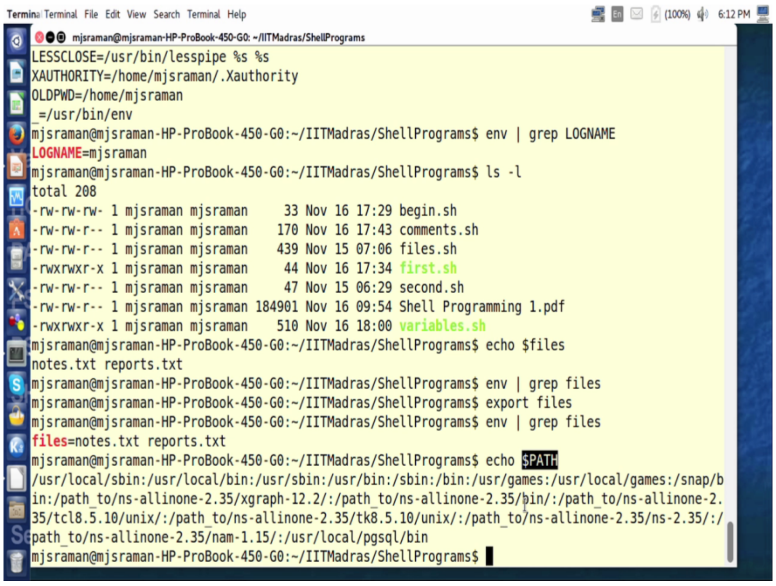
Few System Variables

Variable	Description
HOME	Home Directory
PATH	Directories to be searched for executables
PS1	Prompt for interactive shells.
PS2	Prompt for multiline command
SHELL	Login shell environment

See files.sh for complete demonstration

Now there are few system variables that are there for example Home is the Home directory I think we have seen it, and then path is the directories that are to be searched for any executable code. This is very important because when you are installing lot of programs ok most of the time what happens is that you do not put the path where the binary files are and so even though the binary files will be there in that particular directory you will get an error saying that file not found, and you will be wondering what and finally they will say that set the path variable.

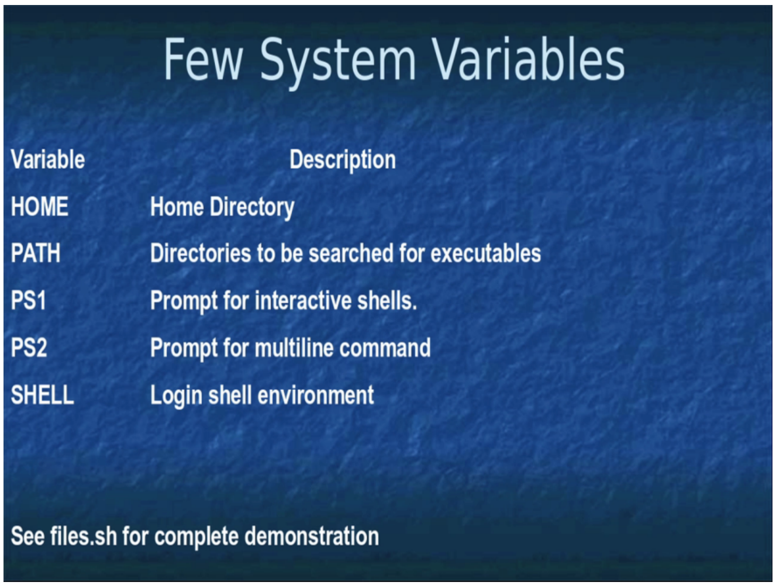
(Refer Slide Time: 07:10)



```
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/mjsraman/.Xauthority
OLDPWD=/home/mjsraman
_=/usr/bin/env
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep LOGNAME
LOGNAME=mjsraman
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ ls -l
total 208
-rw-rw-rw- 1 mjsraman mjsraman     33 Nov 16 17:29 begin.sh
-rw-rw-r-- 1 mjsraman mjsraman    170 Nov 16 17:43 comments.sh
-rw-rw-r-- 1 mjsraman mjsraman    439 Nov 15 07:06 files.sh
-rwxrwxr-x 1 mjsraman mjsraman     44 Nov 16 17:34 first.sh
-rw-rw-r-- 1 mjsraman mjsraman     47 Nov 15 06:29 second.sh
-rw-rw-r-- 1 mjsraman mjsraman 184901 Nov 16 09:54 Shell Programming 1.pdf
-rwxrwxr-x 1 mjsraman mjsraman    510 Nov 16 18:00 variables.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $files
notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ export files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
files=notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/b
in:/path_to/ns-allinone-2.35/xgraph-12.2/:/path_to/ns-allinone-2.35/bin/:/path_to/ns-allinone-2.
35/tcl8.5.10/unix/:/path_to/ns-allinone-2.35/tk8.5.10/unix/:/path_to/ns-allinone-2.35/ns-2.35/:/
path_to/ns-allinone-2.35/nam-1.15/:/usr/local/pgsql/bin
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$
```

So let us take a small example let us spend some time on this , so if you look because this is very important variable that you should understand so echo path for example in this case is in my machine if the binaries are found in any of these paths then it will automatically execute ok? If the binaries are not found in any of this path then it will not execute ok? Not only binaries any executable files. So this is one of the very important variables that you have ok you will be quite often using it and basically shell scripts are some time useful for installation of programs and once you install the program you have to set this path variable otherwise your binaries will you cannot identify and the program might give some errors. This is the binary files are not there.

(Refer Slide Time: 08:00)



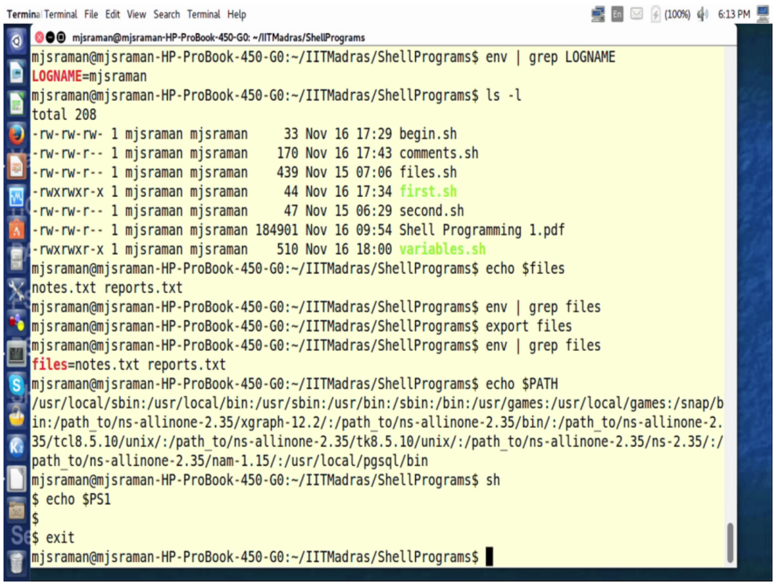
Few System Variables

Variable	Description
HOME	Home Directory
PATH	Directories to be searched for executables
PS1	Prompt for interactive shells.
PS2	Prompt for multiline command
SHELL	Login shell environment

See files.sh for complete demonstration

Then PS1 is for prompt of interactive shells I mean what this mean is ok

(Refer Slide Time: 08:09)

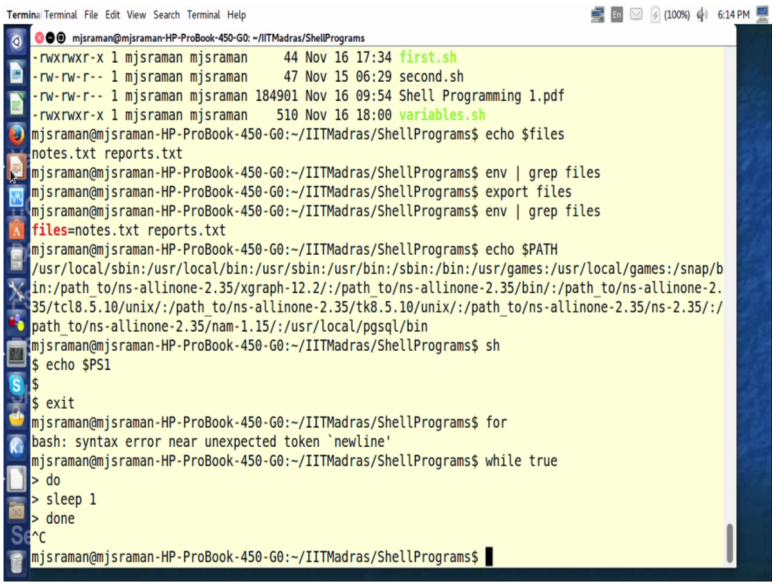


Terminal Terminal File Edit View Search Terminal Help
(100%) 6:13 PM
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms

```
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep LOGNAME
LOGNAME=mjsraman
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ ls -l
total 208
-rw-rw-rw- 1 mjsraman mjsraman     33 Nov 16 17:29 begin.sh
-rw-rw-r-- 1 mjsraman mjsraman    170 Nov 16 17:43 comments.sh
-rw-rw-r-- 1 mjsraman mjsraman    439 Nov 15 07:06 files.sh
-rwxrwxr-x 1 mjsraman mjsraman     44 Nov 16 17:34 first.sh
-rw-rw-r-- 1 mjsraman mjsraman     47 Nov 15 06:29 second.sh
-rw-rw-r-- 1 mjsraman mjsraman 184901 Nov 16 09:54 Shell Programming 1.pdf
-rwxrwxr-x 1 mjsraman mjsraman    510 Nov 16 18:00 variables.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $files
notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ export files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
files=notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/b
in:/path_to/ns-allinone-2.35/xgraph-12.2/:/path_to/ns-allinone-2.35/bin/:/path_to/ns-allinone-2.
35/tcl8.5.10/unix/:/path_to/ns-allinone-2.35/tk8.5.10/unix/:/path_to/ns-allinone-2.35/ns-2.35/:/
path_to/ns-allinone-2.35/nam-1.15/:/usr/local/pgsql/bin
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ sh
$ echo $PS1
$
$ exit
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ 
```

So if you look at this when I am doing this I have this I mean this is the PS 1 variable so if you look at this my PS 1 variable can interpret like this and finally this is the command point where it take the commands. So if I have another shell ok so if you see that my PS 1 variable has changed so echo if I say echo ps 1 then it has become a dollar now i have a shell within a shell ok? So probably yo can export PS 1 and then see what happens ok? You can take it as an exercise.

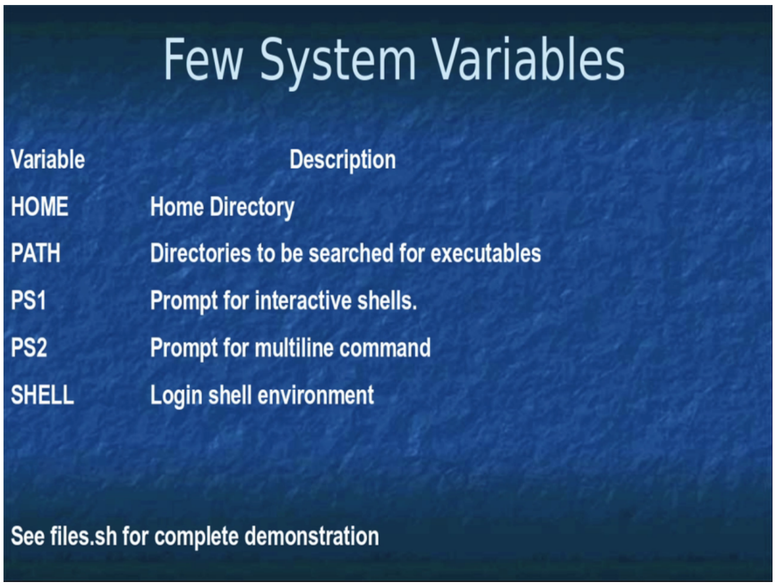
(Refer Slide Time: 08:45)



```
-rwxrwxr-x 1 mjsraman mjsraman     44 Nov 16 17:34 first.sh
-rw-rw-r-- 1 mjsraman mjsraman     47 Nov 15 06:29 second.sh
-rw-rw-r-- 1 mjsraman mjsraman 184901 Nov 16 09:54 Shell Programming 1.pdf
-rwxrwxr-x 1 mjsraman mjsraman    510 Nov 16 18:00 variables.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $files
notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ export files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
files=notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/b
in:/path_to/ns-allinone-2.35/xgraph-12.2/:/path_to/ns-allinone-2.35/bin/:/path_to/ns-allinone-2.
35/tcl8.5.10/unix/:/path_to/ns-allinone-2.35/tk8.5.10/unix/:/path_to/ns-allinone-2.35/ns-2.35/:/
path_to/ns-allinone-2.35/nam-1.15/:/usr/local/pgsql/bin
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ sh
$ echo $PS1
$
$ exit
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ for
bash: syntax error near unexpected token `newline'
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ while true
> do
> sleep 1
> done
^C
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$
```

So now coming back there is another variable called PS 2 which actually for example it expects some command to be given so something like this let us give something like this then this is the PS 2 variable what it says is that I am expecting some more correct commands from you thats what it says ok? So once I type the current it says still your command is not completed. So let us say I put sleep 1 and then I do a done so now it says ok fine it hangs so what is the reason it is hanging? Any idea? okay so I am coming out of this we will see why it does all this ok So the idea here is to show you that if I have something more to be done ok? Then you get this prompt you know the greater than prompt and that is what is given by PS 2.

(Refer Slide Time: 09:45)



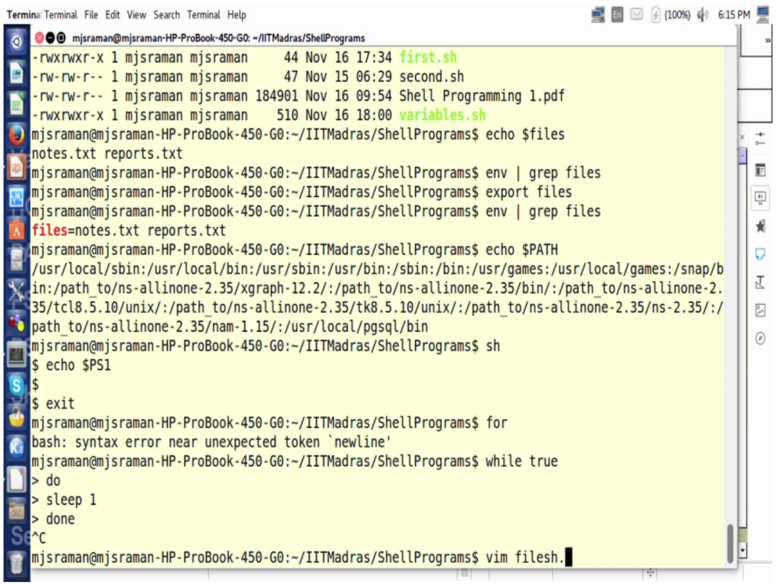
Few System Variables

Variable	Description
HOME	Home Directory
PATH	Directories to be searched for executables
PS1	Prompt for interactive shells.
PS2	Prompt for multiline command
SHELL	Login shell environment

See files.sh for complete demonstration

Prompt for multi line command : So I will type the command in one line that command has not ended therefore the shell is expecting more and more commands and therefore it is giving like this ok? And we already saw what the shell is?

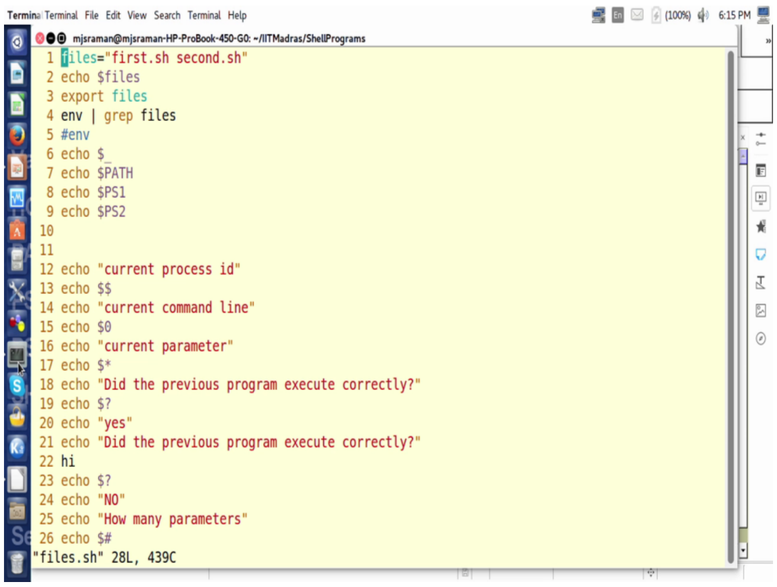
(Refer Slide Time: 10:00)


Terminal Terminal File Edit View Search Terminal Help
(100%) 6:15 PM
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms

```
-rwxrwxr-x 1 mjsraman mjsraman     44 Nov 16 17:34 first.sh
-rw-rw-r-- 1 mjsraman mjsraman     47 Nov 15 06:29 second.sh
-rw-rw-r-- 1 mjsraman mjsraman 184901 Nov 16 09:54 Shell Programming 1.pdf
-rwxrwxr-x 1 mjsraman mjsraman    510 Nov 16 18:00 variables.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $files
notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ export files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
files=notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/b
in:/path_to/ns-allinone-2.35/xgraph-12.2/:/path_to/ns-allinone-2.35/bin/:/path_to/ns-allinone-2.
35/tcl8.5.10/unix/:/path_to/ns-allinone-2.35/tk8.5.10/unix/:/path_to/ns-allinone-2.35/ns-2.35/:/
path_to/ns-allinone-2.35/nam-1.15/:/usr/local/pgsql/bin
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ sh
$ echo $PS1
$
$ exit
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ for
bash: syntax error near unexpected token `newline'
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ while true
> do
> sleep 1
> done
^C
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim filesh.
```

So let us take a look at this file called files dot sh and then we will take a look at how we can get all these concepts together.

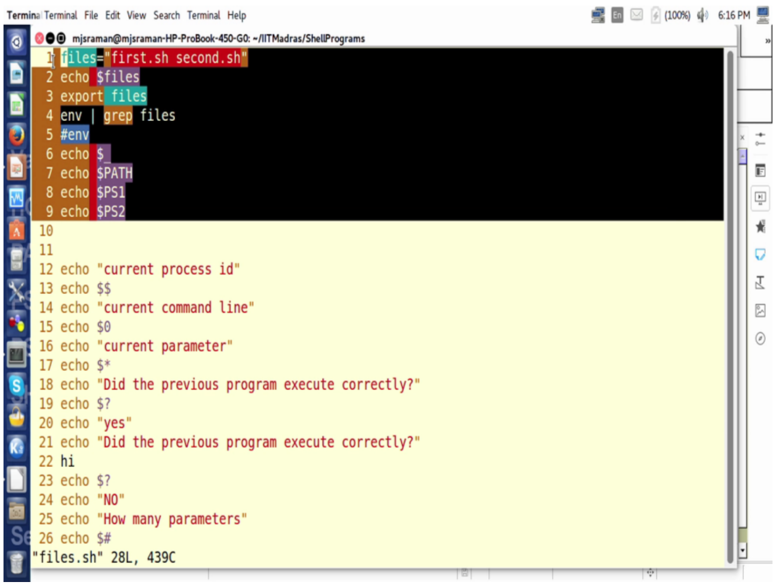
(Refer Slide Time: 10:10)


Terminal File Edit View Search Terminal Help
6:15 PM
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms

```
 1 files="first.sh second.sh"
 2 echo $files
 3 export files
 4 env | grep files
 5 #env
 6 echo $_
 7 echo $PATH
 8 echo $PS1
 9 echo $PS2
10
11
12 echo "current process id"
13 echo $$
14 echo "current command line"
15 echo $0
16 echo "current parameter"
17 echo $*
18 echo "Did the previous program execute correctly?"
19 echo $?
20 echo "yes"
21 echo "Did the previous program execute correctly?"
22 hi
23 echo $?
24 echo "NO"
25 echo "How many parameters"
26 echo $#
"files.sh" 28L, 439C
```

So here is the first line of the code, so what it does is if you remember I had executed this and shown you that it works ok then the fifth line is a comment ok? And then I have something like echo dollar underscore echo dollar path, dollar PS 1 and dollar PS 2 I think we know all these echo dollar path will give you what all the path for the executables and let us see what this echo dollar underscore says because this is something new and as I told you there are lot of funny characters that come in shell programming so this is one such character ok so let us look at what it does ok so let us just try to do echo dollar underscore ok, so it says files dot sh now what is it ok, take this as a next case and find out what does echo dollar underscore means ok so let us go and find out files dot sh ok.

(Refer Slide Time: 11:15)

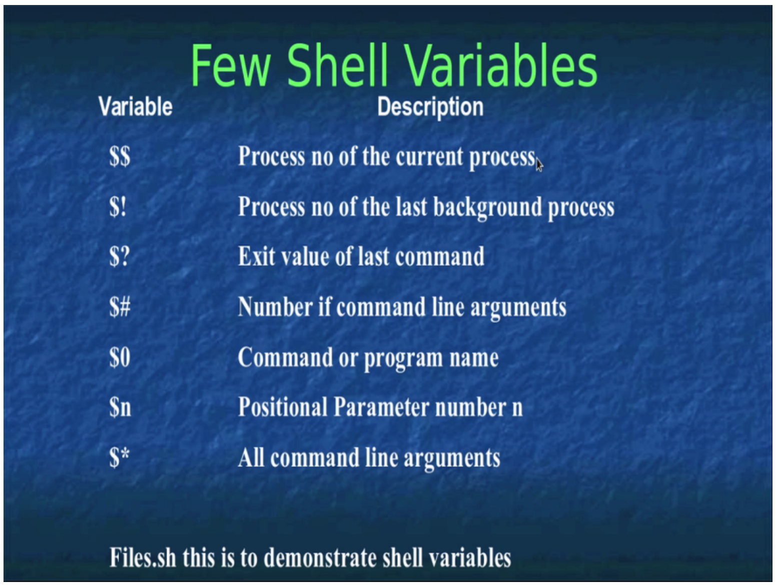

Terminal File Edit View Search Terminal Help
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms

```
files="first.sh second.sh"
echo $files
export files
env | grep files
#env
echo $
echo $PATH
echo $PS1
echo $PS2


echo "current process id"
echo $$
echo "current command line"
echo $0
echo "current parameter"
echo $*
echo "Did the previous program execute correctly?"
echo $?
echo "yes"
echo "Did the previous program execute correctly?"
hi
echo $?
echo "NO"
echo "How many parameters"
echo $#
"files.sh" 28L, 439C
```

And then you are echo dollar path echo dollar ps 1 echo dollar PS 2 so these are some of the environmental variables that you have let us move on to one of the other kinds of variables ok?

(Refer Slide Time: 11:29)



Few Shell Variables

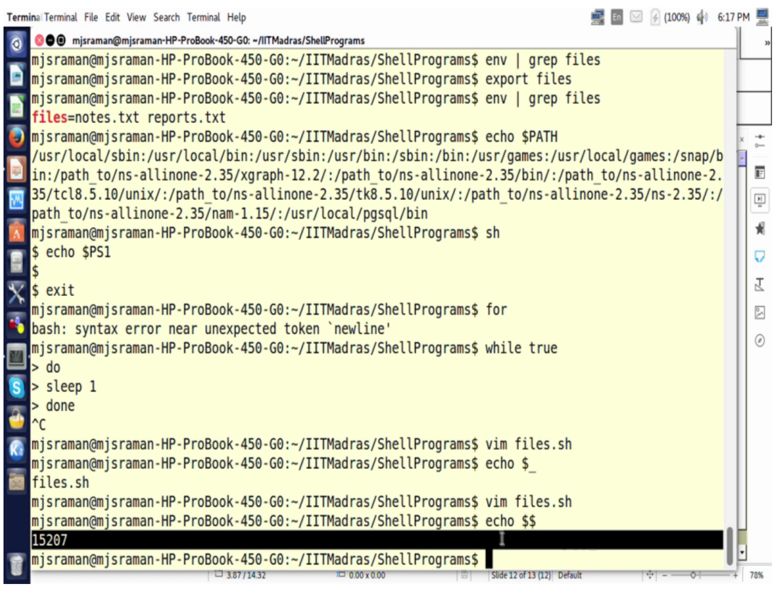
Variable	Description
$$	Process no of the current process
$!	Process no of the last background process
$?	Exit value of last command
$#	Number if command line arguments
$0	Command or program name
$n	Positional Parameter number n
$*	All command line arguments

Files.sh this is to demonstrate shell variables

we saw the environmental variables and the system variable can be exported etc now moving on , ok we also have something known as shell variables this is very important the reason why this is important is man of the times it see for example in C programming language you have something how to read arguments from the user from a program so for example input arguments we know how to print but I told you that you cannot be expecting user to send you all the data it cannot be an interactive progams so if it is not an interactive program what happens is the user has to type in the command line what are values he wants to pass now how do you interpret those values. So for that the shell provides some built in variables so let us look at some of built in variables so if you put dollar dollar then it tells you that what is the process number of the current process I mean I hope you are very familiar now with what is process numbers. See process ID is something unique to every process that is running in the operating system.

So if you want to know what is the process ID of the current process I can use this dollar dollar command.

(Refer Slide Time: 12:45)

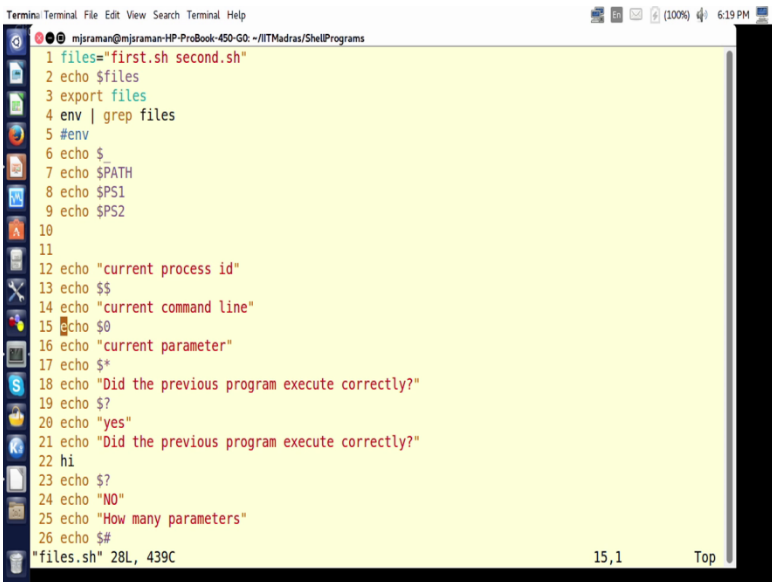

Terminal File Edit View Search Terminal Help
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms

```
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ export files
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ env | grep files
files=notes.txt reports.txt
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/b
in:/path_to/ns-allinone-2.35/xgraph-12.2/:/path_to/ns-allinone-2.35/bin/:/path_to/ns-allinone-2.
35/tcl8.5.10/unix/:/path_to/ns-allinone-2.35/tk8.5.10/unix/:/path_to/ns-allinone-2.35/ns-2.35/:/
path_to/ns-allinone-2.35/nam-1.15/:/usr/local/pgsql/bin
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ sh
$ echo $PS1
$
$ exit
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ for
bash: syntax error near unexpected token `newline'
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ while true
> do
> sleep 1
> done
^C
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim files.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $_
files.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim files.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $$
15207
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$
```

So if you look at this if you I can even type it like this so if I put echo dollar dollar so it shows 15207 which is the process ID of the current shell that it is running. Now let us go back and then see so next I have dollar ban. So this tells you the process number of the last background process so we know clearly that we can run a process in a foreground or a background and so if you want to know the background process number you can use this. Now the third one is very important now this tells you whether your previously executed command is successful or not see UNIX has a convention that if your return value is 0 then the previous program has executed successfully any other value other than 0 then your program has not executed successfully so this you can identify I mean as I told you shell scripting allows you to take decisions based on whether a command has run successfully or not. So to take that decision this will be very useful we will see the usage of this.

And dollar hash tells you the number of command line argument so let us take a look at this if you look at this, let us go to the files dot files dot sh.

(Refer Slide Time: 14:14)

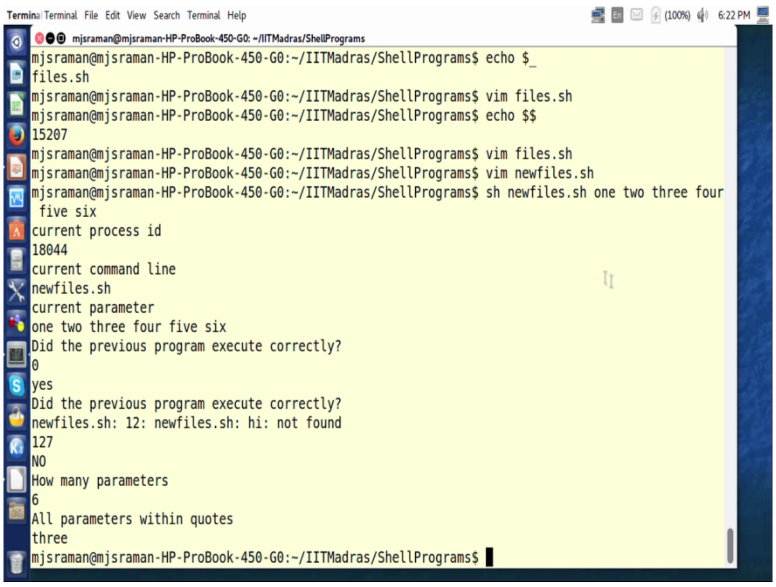

Terminal Terminal File Edit View Search Terminal Help
mjsraman@mjsraman-HP-ProBook-450-G0: ~/IITMadras/ShellPrograms
(100%) 6:19 PM

```
files="first.sh second.sh"
echo $files
export files
env | grep files
#env
echo $_
echo $PATH
echo $PS1
echo $PS2


echo "current process id"
echo $$
echo "current command line"
echo $0
echo "current parameter"
echo $*
echo "Did the previous program execute correctly?"
echo $?
echo "yes"
echo "Did the previous program execute correctly?"
hi
echo $?
echo "NO"
echo "How many parameters"
echo $#
"files.sh" 28L, 439C                                          15,1          Top
```

So if you look at this as I told you you can have a current process ID you can have a current umm so if you say dollar hash it tells you the number of command line parameters. So let us and if you look at the other option, we will first go through all the options and then we will come back then if you say dollar 0 then it gives you the command or the other program name isn't she the very first word that you typed after the prompt I mean the dollar is called as the prompt in the shell script we had that prompt PS 1 is the prompt and dollar star gives all command line arguments and dollar n gives the position parameter let me demonstrate it with a small program. So that you understand this.

(Refer Slide Time: 14:58)

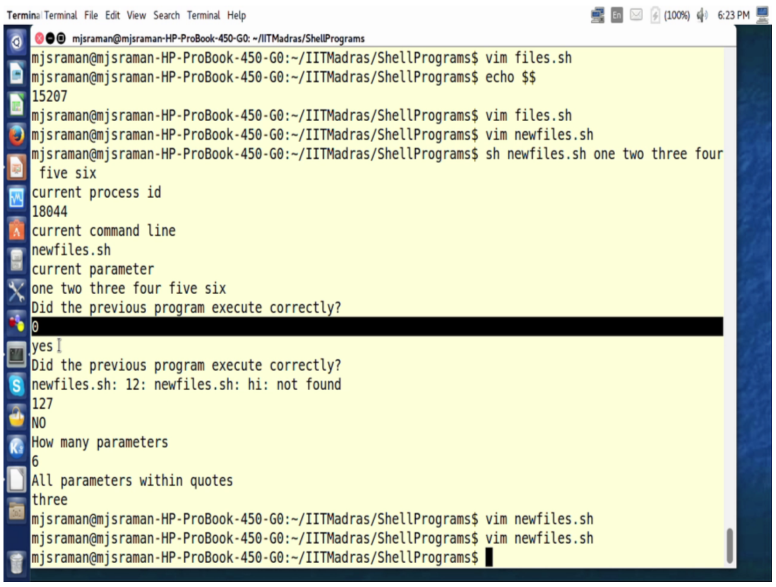


```
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $_
files.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim files.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $$
15207
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim files.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim newfiles.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ sh newfiles.sh one two three four
 five six
current process id
18044
current command line
newfiles.sh
current parameter
one two three four five six
Did the previous program execute correctly?
0
yes
Did the previous program execute correctly?
newfiles.sh: 12: newfiles.sh: hi: not found
127
NO
How many parameters
6
All parameters within quotes
three
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$
```

So if you look at this program the first part of it we all saw how to I will make a copy of this program and then we will, so I can just make it as say a new files dot sh , I will take new files dot sh and then I will remove this part because we all quite well understand the PS 1 and other kind of shell variables, what we will now do is we will now try to run this program so let us look at this in line number 2 I can see what is the current process ID in line number 5 it will print what is the current command that is being executed in line number 6 and 7 it tells you what are the current parameters and if you want to know whether this line has executed correctly then I have to do a echo dollar so if it is echo dollar question mark becomes 0 then the current the previous program has executed correctly. So for example here I am making a deliberate mistake if you see this. So because I am making a deliberate mistake I expect that this value should not be 0 .

Whatever is getting printed should not be 0 and so I am printing echo "NO" and then if I want to know how many parameters I am passing to this program when I can put dollar hash and I can just say what is the third parameter that I am passing so let me just run this program and show you demonstration if you look at this program the way I have to run it is I can say new files dot sh and then I can give some parameters 1 2 3 4 etc 5 6 etc. Now what I have typed is sh new files 1 2 3 4 5 6, now look at this the first current process ID is 18044 which is cool. Now coming back it says current command line so let us look at the code what is the current command line that it is printing.

(Refer Slide Time: 17:05)



```
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim files.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ echo $$
15207
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim files.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim newfiles.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ sh newfiles.sh one two three four
 five six
current process id
18044
current command line
newfiles.sh
current parameter
one two three four five six
Did the previous program execute correctly?
0
yes
Did the previous program execute correctly?
newfiles.sh: 12: newfiles.sh: hi: not found
127
NO
How many parameters
6
All parameters within quotes
three
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim newfiles.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$ vim newfiles.sh
mjsraman@mjsraman-HP-ProBook-450-G0:~/IITMadras/ShellPrograms$
```

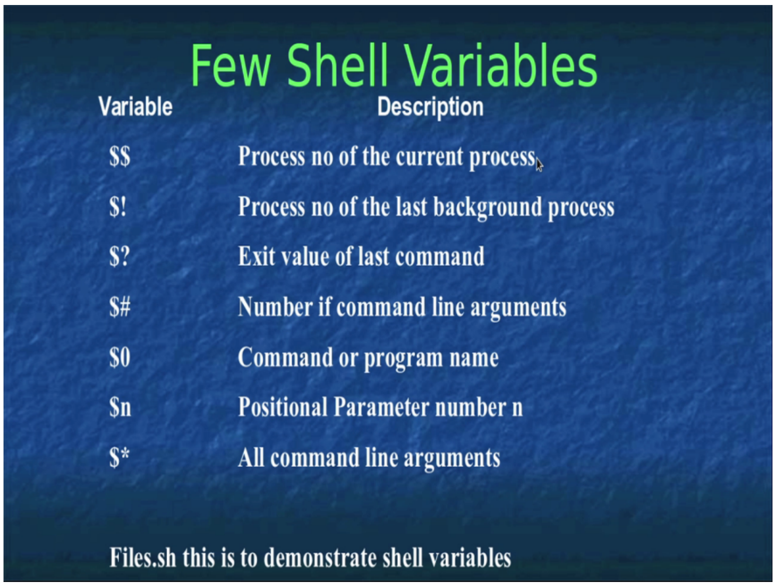
So if you look at new files that is currently dollar 0 so if you look at dollar 0 what it prints is the new files dot sh. So this is considered to be dollar 0. The current command that you are running is considered to be the 0 output it is known as the positional parameter. And then what are all the current parameters so since this is becomes the file name then whatever you are passing after this becomes the parameter for this file. So what happens here is what are the current parameters you are passing is 1 2 3 4 5 6. So how do I print it ? So if you look at this file I print it Via dollar star. So when I put echo dollar star it prints 1 2 3 4 5 6. Now what I willl do is I am trying to see whether the previous program execute correctly just a question so this echo should print it correctly the this should pass. So let us see whether it is got printed correctly yes it has got printed correctly and the program returned correctly therefore it is printing a value of 0.

Then again I am printing yes. Now let us come to this line look at this I typed hi and hi is not a command in UNIX. Therefore what it chases the command not found and look at the return value I have pressed I actually typed echo dollar question mark and if you see this it returned the value 127 which is not 0. Please understand that. This is one of the reasons in C programming language they say that if you do anything correctly return a 0.

So if you look at this so 127 as I told you any value greater than 0 tells you that the previous command not execute correctly. So here the hi command was not found therefore the shell did not execute this correctly therefore it returned the value of 127 and therefore I have printed no then how many parameters then I am putting dollar hash. This tells you the total number of parameters that are passed for this program.

So if you are very familiar with C programming language you have int arg c then arg v. So the arg c actually tells you how many parameters are getting passed so in this case you can do a dollar hash and find out how many parameters are getting passed. And then I can find out the particular parameter by giving dollar and a number after this so in our case 1 2 and 3. So essentially we will be seeing that 3 the value three will be printed in the last line so look at this so the value three is getting printed.

(Refer Slide Time: 19:54)



Few Shell Variables

Variable	Description
$$	Process no of the current process
$!	Process no of the last background process
$?	Exit value of last command
$#	Number if command line arguments
$0	Command or program name
$n	Positional Parameter number n
$*	All command line arguments

Files.sh this is to demonstrate shell variables

So this gives you brief so these are all the few shell variables these are very very important when you are writing shell scripts you will be using them quite a lot so type some exercises and try to get control of what these variables actually mean. So with this we come to the end of this session and we will continue with sequencing and others flow from the next session onwards. Thank You!