Institute for Advanced Studies
in Basic Sciences
Gava Zang, Zanjan, Iran

First NIGS workshop on

Bash Programming for Geophysicists

Abdolreza Ghods

Institute of Advanced Studies in Basic Sciences, IASBS, Iran

Version 1.0 June 2013

A. Ghods aghods@iasbs.ac.ir

Why Bash Scripting?

As geophysicists , we are frequently forced to

1) Obtain quick statistics from an existing large data bank,
2) Extract the required data from a large volume of data,
3) Convert automatically large volume of data to different formats (e.g., convert SEED seismic waveforms to SAC),
4) Do sensitivity analysis by running a given program for different possible range of input parameters,
5) Draw complex GMT plots,
6) Do parallel computing

A. Ghods aghods@iasbs.ac.ir

What is bash programming?

- It is a programming language but a simple one,
- It is a command-line interpreter and does not need any compilation step,
- All Linux commands are available for bash programming,
- If the routine Linux commands are not enough, it has its *own variables, do loops, if* and other compact programming tools like *`awk` and `sed`*
- The power comes when it glues easily all Linux commands with others programs including those of yours.

A good site to learn more about bash programming http://www.tldp.org/LDP/abs/html/

A. Ghods aghods@iasbs.ac.ir

Step 1
Simple Linux commands

- In this section, I will show you the power of simple individual Linux commands. Learning these commands is very easy but essential to write bash scripts.
- To do the examples in this tutorial, please copy the directory `bash_scripting` to your home directory.

A. Ghods aghods@iasbs.ac.ir

Quick statistic for files in a data bank?

- As Geophysicists, we observe earth parameters and keep our observations in large data banks which often are made of large number of files. For example, continuous seismic waveforms from different stations are recorded in different directories under name of each station. Each station directory holds different subdirectory for each year. Each year subdirectory is consisted of 12 month subdirectory. Continuous waveform data is kept in each month subdirectory usually as separate 30 minutes files. For a single station for one year period, we are dealing with 365 by 48 individual files! We should be able to extract our required statistics, and data from such large data bank!

A. Ghods aghods@iasbs.ac.ir

Eaxmple 1-1

Make a list of files ls

- Once you have copied `bash_scripting` directory into your home directory, use *`cd`* command to go to the directory and then use the command *`ls`* to see its contents.
- You can return to your home directory either by executing command `cd ..` or `cd ~` or `cd $HOME`.
- Please notice that all linux commands are case sensitive so ls is not equivalent with Ls or LS !

A. Ghods aghods@iasbs.ac.ir

Example 1-2

ls and *wild cards*

- We often have large number of files within one given directory and want to find those who follow a specific pattern. For example in the directory `~/bash_scripting/example1/BHRC` we want to find those nordic waveform files that are recorded by only two stations. This means we want those files which have `006` at the end of file. To find the file we use the following command,

  `ls *006`

  `*` sign is one of the wild character and means whatever file which ends with `006`

A. Ghods aghods@iasbs.ac.ir

Example 1-2 continued

- Now if we want to find only waveforms of year 1990 which are recorded by two stations, we use following command. Each station has three components.

```
ls 1990*006
```

- If we want to know the list of waveforms recorded by one or three stations, we use following command

```
ls *00[3-9]
```


A. Ghods aghods@iasbs.ac.ir

Example 1-2 continued

- If we want to have a list of waveforms belong to month 06, we use following command

  `ls ????-06-*`

  notice that the command `ls *06` is not equivalent to ls `????-06-*`!

A. Ghods aghods@iasbs.ac.ir

Example 1-3

Sorting files in terms of size

- In some cases, the size of files gives some critical information for our work. For example, if we have separate bulletin and phase files for different events and want to locate only events with large number of phase readings, we choose only those files with larger size. So the easy method to choose the proper files is to sort the files in terms of their size.

A. Ghods aghods@iasbs.ac.ir

Example 1-3 continued

Directory `~/bash_scripting/mloc` contains a number of bulletin and phase info files along with other related files for a cluster of events in Zagros. To sort the files in terms of their size in descending order, we use following command,

```
ls -l -S *ffb
```

the switch `-l` gives more information about each file including its size and time of creation and its permission. The switch `-S` makes ls to sort the files in descending size.

A. Ghods aghods@iasbs.ac.ir

Example 1-4

Sorting files in terms of their modification time

- Sometimes, we would like to know the latest files that we copied in a given directory or we modified. For example, if we would like to know what pdf files are among our latest downloads, we use following command,

```
ls -t -l ~/Downloads/*pdf
```


A. Ghods aghods@iasbs.ac.ir

Example 1-5

Size of files du

- We often need to know the size of a given directory. For example if we want to know what is the size of directory `BHRC` in directory `example1`, we use the following command,

```
du BHRC
```

- If we want to know how many kb is the size of the `ffb` files in directory `mloc`, we use following command. The command gives size of each file and then at the end produces a grand total.

```
du -c *ffb
```


A. Ghods aghods@iasbs.ac.ir

Example 1-6

Copying files cp

- We often need to transfer our data from external storage to our hard disks or vise versa. The easiest way to copy large number of directories or files is to use command line command `cp`. For example to copy the contents of this course to your home directory, use the following command,

  ```
  cp -r /media/mydisk/bash_scripting ~
  ```

  the switch `-r` makes the `cp` command to copy a directory and all of its subdirectories.

A. Ghods aghods@iasbs.ac.ir

Example 1-7

Erasing files rm

- We always need to erase files which we do not need anymore. In this example we first copy all `ps` files inside the `mloc` directory using following command,

  ```
  cp mloc/*ps .
  ```

  Then use following command to erase all of the `ps` files,

  ```
  rm *ps
  ```

Example 1-7

Erasing a directory rm

- We always need to erase directories of files which we do not need. In this example first copy directory `mloc` using following command,

  ```
  cp -rf mloc mloc2
  ```

  Then use following command to erase directory mloc2

  ```
  rm -rf mloc2
  ```

  be careful in using `rm -rf` because this command without asking you will erase all files inside a given directory including all of its subdirectories.

A. Ghods aghods@iasbs.ac.ir

Example 1-8

Moving files or directories mv

- We always need to reorganize our directories or files around our file system or rename a file or directory. You can rename the directory `test` under `~/bash_scripting/example1` to `test2` using following command,

  `mv test test2`

- You can move directory `test2` to directory `moved` using following command

  `mv test2 moved`

  To see if `test2` is inside directory `moved`, use command *`ls`*.

File management in Linux

- Linux manages everything as files and holds different things in different directories. The / directory is called root directory. You can see all main directories of Linux by using command `ls /` . The most important Linux directories are

  `/bin /home /usr /opt /var /tmp /lib /sbin /etc`

A. Ghods aghods@iasbs.ac.ir

File permissions in Linux

- Linux is very safe once compared to weak operating systems such as windows -:(
- The root of this strength is its extended file permissions. Each file has read, write and execute permission for three different group of users, namely the user, the group user belong to it and other user groups. Let's explore these permission using command `ls -l`

A. Ghods aghods@iasbs.ac.ir

Example 1-9

Changing permission of a file or directory chmod

- To change permission of a given file we use command `chmod`.

  `chmod +x file` : add executable permission

  `chmod 700 file` : gives full permission to the user and no permission to others.

  `chmod 755 file`: gives full permission to the user and only read and execute permission to the others.

- To apply a given permission to all files within a given directory, use `chmod -r directory_name`

A. Ghods aghods@iasbs.ac.ir

Example 1-10

Changing ownerships of a file or directory chmod

- Each file and directory on Linux file system belong to a given user and group. To change the ownership of a file use command

```
chown user.group name_file
```

- To change ownership of all files and subdirectories under a given directory use command `chmod -R`

A. Ghods aghods@iasbs.ac.ir

Example 1-11

Display current Linux tasks top

- It is very important to know how many jobs or tasks are being run by your Linux machines. Sometime your Linux box gets stupidly very slow. To find why, the first thing to do is to know what are the running tasks and if any of them are using large cpu or large memory. To see the list of tasks and their specifications, use the command `top`. Once you found the buggy program using large memory or cpu, you can kill it using command `kill PID_number_of_the_job`.

A. Ghods aghods@iasbs.ac.ir

Example 1-12

Display file system disk space usage df

- Before running a large time consuming program which produces large volume of output, or starting to copy large volume of data into a hard disk, it is always wise to check how much disk space is available on your hard disks. The command `df` will give these information for all of your hard disks and their partitions.

A. Ghods aghods@iasbs.ac.ir

Example 1-13

Display who is logged on a system who

- Linux is a full truly multi-users operating system and a given Linux machine can host several users simultaneously. If you are working on a Linux client or server, you can check who are currently logged into the system using the command *who*

A. Ghods aghods@iasbs.ac.ir

Example 1-14

Connecting to other computers ssh

- If you have an account, you can login into a given linux machine remotely using command `ssh`

  `ssh aghods@iasbs.ac.ir`

  `ssh aghods@192.168.14.60`

- *You can transfer files between different linux computers using command* `scp`.
- `Telnet` and `ftp` are other ways of remote communications in Linux but these are not secure and are not recommended.

Example 1-15

Display the on-line manual pages man

- Anytime you want to know more about a given command without searching the internet, use the master command `man`. For example to know all switches of *`ls`* command use following command,

  `man ls`

A. Ghods aghods@iasbs.ac.ir

Example 1-16

Finding files locate and find

- If you forgot where is a given file, the best way is to use command `locate`. For example `locate bash` will find all files and directories which have the `bash` word in them.
- *`locate`* command is very fast in finding files which are about one day old. `locate` is based on a data bank which is updated every night by a scheduled task. If your computer is off during the night all the time so `locate` does not work. To avoid this problem, you can use command *`find`*. This works all the time but it is slower.

A. Ghods aghods@iasbs.ac.ir

Example 1-17

Packing and unpacking files tar

- If we have a data bank consisting of many files, the wisest way is to pack all directories and files to a single `tar` file. Please notice that `tar` command does not reduce the total size of the files but only bundle them into a one single file. As geophysicists, we often receive data and softwares from others as bundled tar files. To unpack them we also need `tar` command.

  Examples:

```
tar -cf archive.tar foo bar  # Create archive.tar from files foo and bar.
tar -tvf archive.tar         # List all files in archive.tar verbosely.
tar -xf archive.tar          # Extract all files from archive.tar.
```


A. Ghods aghods@iasbs.ac.ir

Example 1-18

Compressing files gzip, gunzip

- To reduce size of our data bank we can compress the files using command `gzip. gzip` is very successful in reducing size of text file and earthquake waveform files.
- Many data and software distributions are in the form of gzipped file. To unzip a given gzipped file use command `gunzip`.
- You can also compress and bundle simultaneously a set of files using command `zip`. To `unzip` a compressed zip file use command `unzip`.

A. Ghods aghods@iasbs.ac.ir

Other useful Linux commands

```
more
head
tail
cat
pwd
grep
who
mkdir
```

```
date
at
time
sort
uniq
telnet
ftp
rsync
```


A. Ghods aghods@iasbs.ac.ir

Exercise 1

- Print the first 10 lines of file `20071130.1332.ffb` in the directory `example1/mloc`. You should use command `head` with the proper switch to do the above task. Use command `man` to know more about command `head`.
- Make a tar gzipped file from contents of `~/bash_scripting` directory.
- What is the total size of the `example1/mloc` directory.
- Copy all `ffb` files in directory `example1/mloc` directory to `/tmp` directory
- Copy `example1/mloc` directory to `/tmp` directory
- Rename `/tmp/mloc` to `/tmp/mloc2`
- Change permissions of `/tmp/mloc2` directory so everybody from all groups can only read the files.

A. Ghods aghods@iasbs.ac.ir

Step 2
Combining several Linux commands

- In this step, we will learn more commands but we also try to combine several commands to do a given task.

A. Ghods aghods@iasbs.ac.ir

The *grep* command

Find a word within a large file

- In many cases, we would like to know how many times a word has been used in a given text file. For example in seismology, we need to need to know how many phases a given station has in a given catalog, or we would like to know how many Pn and Pg phases we have in a given catalog. A quick way to find such statistics is to use the `grep` command.

A. Ghods aghods@iasbs.ac.ir

Example 2-1: The *grep* command

- To find number of Pg phases in file `collect.out` given in directory `example1`, we run command `grep “Pg “ collect.out` on the command line,

```
[aghods@ghods example1]$  grep "Pg " collect.out
 PRN  GZ EPg  9 D 1850  3.25                          117    0.06 0 29.6 242
 ALA  GZ EPg    C 1850  3.70                          113   -0.4610 35.4 153
 FIR  GZ EPg    D 1850 11.17                           93    0.1010 81.3 172
 SHM  GZ EPg    D 1850 12.05                           93    0.2110 86.1 136
 .....
```

The number of lines with Pg word is large so I did not show all of them here. Please notice that *[aghods@ghods example1]$* is not part of the command and it is part of command line and its form depends on your setup. You should only type the command (grep “Pg “ collect.out) in the command prompt to get above result.

A. Ghods aghods@iasbs.ac.ir

Example 2-2: The *wc* command

- As a geophysicist, in many cases we need to know how many lines are inside a given file. To find number of lines inside `collect.out` file inside the directory `example1`, we use command *`wc -l collect.out`*

```
[aghods@ghods example1]$ wc -l collect.out
2079 collect.out
```

The command shows that the `collect.out` file has 2079 lines. The *`-l`* after command *`wc`* is called a switch. Majority of Linux commands accepts switches which somehow modify their behavior The *`-l`* switch makes *`wc`* to prints only number of lines and ignore printing other info like number of words in the document.

A. Ghods aghods@iasbs.ac.ir

Example 2-3: Redirection >

- In Example 1-1, we found number of lines having “Pg “ words and printed them into the screen. The number of lines were large so they ran out of the screen and we can not easily count them. To count the number of Pg phase lines, we first redirect output (>) to temporary file called `junk` and then count number of lines inside file `junk` using command `wc`.

```
[aghods@ghods example1]$ grep "Pg " collect.out > junk
[aghods@ghods example1]$ wc -l junk
352 junk
```

We can see that 352 Pg phases exist inside the `collect.out` catalog file.

Example 2-3

continued Redirection >>

- If we would like to add output of a given command to the end of a preexisting file, instead of > we should use >>. If we use > this will first erase the contents of the file and then write the output of a command to the file.

A. Ghods aghods@iasbs.ac.ir

Example 2-4
Piping |

- A better way to find number of Pg phase lines is to pipe the output of `grep` command directly to `wc` command. This powerful action is called piping and can be done infinitely.

```
[aghods@ghods example1]$ grep "Pg " collect.out | wc -l
352
```

- Again the combined command produced the same result of 352 Pg phase lines.

A. Ghods aghods@iasbs.ac.ir

Example 2-5

Running job in the background *nohup*

- Many times we would like to run a large time consuming job on a Linux workstation without being logged in after running the job. To run a program in the background (without being logged in), the program should be run by `nohup` command. At this stage, we assume that the program does not ask for any input interactively.

```
nohup program1 &
```

- To see if the program is running, use the command `top`. The `nohup` command directs automatically the output of the `program1` to file `nohup.out`

A. Ghods aghods@iasbs.ac.ir

Example 2-5
continued *nohup*

- If our program asks interactively for a set of input parameters, we should first write each parameter in separate lines within a file (here `input.dat`) and then use the following command,

```
nohup program1 < input &
```


A. Ghods aghods@iasbs.ac.ir

Example 2-5
continued nohup and matlab

- These days many geophysicist do their computational task using Matlab. If your atlab program is a very time consuming program and you need to run on it as background job, you should do as follow,

```
nohup matlab -nodisplay < YOUR_MATLAB_FILE.m > output.dat &
```

With above command matlab will run without opening a graphical console. Screen output of your matlab program will be directed to file `output.dat`.

A. Ghods aghods@iasbs.ac.ir

Example 2-6

Measuring execution time of your program!

- If you want to know how long running a program will take on your computer, you can use command `time` to measure the time as follow,

```
time ./myprogram
```

once program is finished you will have statistic about the timing of your program.

A. Ghods aghods@iasbs.ac.ir

Exercise 2

- Find number of events in the catalog file `collect.out` using only `grep` and `wc` commands. Notice that lines with 1 in column 80 are event lines.
- Find number of phases recorded by station FIR
- Find number of Pn phases recorded by station FIR
- Find number of Pn phases recorded by component SZ of station FIR

A. Ghods aghods@iasbs.ac.ir

Section 3

- In this section we will learn the powerful *awk* utility of bash scripting. The *awk* command is very essential in all of the simple and complex scripts that we develop in this course.
- Use command `nice` to change priority of running your program

A. Ghods aghods@iasbs.ac.ir

The *awk* command

- *Awk* utility is a line processor program which gives us much more power to extract required information from a large text file like an earthquake catalog file. *Awk* can do simple computation on different columns of a text file. In this way, *awk* resembles *Excel Microsoft*.

A. Ghods aghods@iasbs.ac.ir

Example 3-1: *awk* command

Printing selected number of columns

- We often need to have a subset of columns from a file having many columns. For example in file `example3_awk/brojen.txt`, we need to have latitude and longitude columns (cols 5 and 6). To extract these columns, we do as follow,

```
awk '{print $5, $6}' brojen.txt
```


A. Ghods aghods@iasbs.ac.ir

Example 3-2 : *awk* command
Ignoring header lines

- Most of the time, our data files have text header lines. We count the number of header lines and use switch NR to signal the *awk* command about the lines. File `brojen1.dat` is the original version of file `brojen.dat`. We extract as before the columns 5 and 6 but ignore the header lines,

```
awk 'NR>18 {print $5,$6}' brojen1.txt
```


A. Ghods aghods@iasbs.ac.ir

Example 3-3: *awk* command

The *substr* facility

- It is a common practice to encounter cases where information in different columns are not separated by a space or some other character. For example in file `example3_awk/earthquake_catalog.dat` for events happening in two digit month and two digit days, the column for month stick to the day column. Thus we can not extract column lat and long using the approach in the previous slides. In these cases, we use `substr` facility within `awk` command to extract our desired information. We request our data by specifying the number and width of space columns (not information columns) instead,

```
awk ' {print substr($0,26,17)}'
earthquake_catalog.dat
```


A. Ghods aghods@iasbs.ac.ir

Example 3-4: *awk* command

if and awk

- Often we need to extract a subset of information based on the contents of information presented on different rows in a a data file. For example if we want to extract lat and long and magnitude of events in file `earthquake_catalog.dat` having magnitude larger than 3, we should use `if` in `awk` command as follow,

```
awk '{if (substr($0,76,3) > 3) print
substr($0,26,17),substr($0,76,3)}'
earthquake_catalog.dat |more
```


A. Ghods aghods@iasbs.ac.ir

Example 3-4: *awk* command continued *if and awk*

- If we need only events with magnitude larger than 3 and smaller than 4, we do as follow,

```
awk ' {if (substr($0,76,3) > 3 &&
substr($0,76,3) < 4) print
substr($0,26,17)}' earthquake_catalog.dat
|more
```

- In addition of above conditions, if we need only those events not having fixed depth, we do as follow,

```
awk ' {if (substr($0,76,3) > 3 &&
substr($0,76,3) < 4 && substr($0,55,1) !=
"F") print substr($0,26,17),substr($0,76,3)}'
earthquake_catalog.dat
```


A. Ghods aghods@iasbs.ac.ir

Example 3-4: *awk* command

internal variables *and awk*

- We can reduce the complex `awk` command in previous slide to a simpler form using internal variable inside `awk`. We can introduce a variable called `mag` to store the value of `substr($0,76,3)` as magnitude of an event and refer to this variable instead.

```
awk ' { mag=substr($0,76,3); if (mag > 3 &&
mag < 4 && substr($0,55,1) != "F") print
substr($0,26,17),mag}' earthquake_catalog.dat
```


A. Ghods aghods@iasbs.ac.ir

Example 3-5 a*wk* command

Calculations on columns

- The geographical coordinates of Azerbaijan network is given in file `Azerbaijan_stn.dat` in `example3_awk` directory. We would like to reformat it to hypo77 format which is used by seisan and hypo77. In hypo77, the latitude and longitude are given in degree and minutes. With the following command we do the required computation and reformatting. Please notice that instead of `print`, `printf` or formatted print is used. `printf` in *awk* is exactly similar to that in *C* programming language.

```
awk 'NR >1 {lat=substr($2,1,2);lat2=substr($2,4,3);
long=substr($3,1,2);long2=substr($3,4,3); printf "%5s
%2d%5.2fN %2d%5.2fE%4d \n",
$1,lat,lat2*.06,long,long2*.06, $4}'  Azerbaijan_stn.dat
```


A. Ghods aghods@iasbs.ac.ir

Exercise 3

- Extract bulletin info for events having magnitude larger than 4 using file `collect.out` in `example3_awk`
- Find number of Pg phases with epicentral distances of less than 100 km. (Please notice that you should convert epicenttral distance from character to a real number using `strtonum` facility of `awk`.

A. Ghods aghods@iasbs.ac.ir

Section 4

In this section, we will try to write bash scripts and learn different aspects of bash scripting behind simple Linux commands.

A. Ghods aghods@iasbs.ac.ir

How to make a simple bash script

- We can make a bash script simply by entering our simple Linux commands within a text file and then make the file executable (`chmod +x your_bash_file`) and run it on the command line (`./your_bash_file`)
- Do not forget the ./ ! The ./ indicates that your bash file exists in the current directory.

A. Ghods aghods@iasbs.ac.ir

Example 4-1

Running several tasks consequentially

- We are often forced to do several commands to perform a given task. For example to draw a GMT plot, in most of cases we are forced to run several commands consequently. Directory `example22_color_pallate_for_psxy` in directory `example4` contains an example for GMT script.

A. Ghods aghods@iasbs.ac.ir

Example 4-2

Automation using Do loops

- We would like to write a program which report number of files in all directories start with `D` inside `example_apsheron_irsc_data`. We use Do loops to write the bash script. The bash script is `example4_2_number_dir.bash` in `example_apsheron_irsc_data`.

Example 4-3

Automation `sort` *and* `uniq`

- Now we would like to improve the previous example by reporting the total number of files in the directory `example_apsheron_irsc_data`.

  Here we use again bash variables. We also show how to do integer algebraic calculations using bash variables.
- You can find the bash script at `example4_3_number_dir_sum.bash` in `example4`.

Example 4-4

Automation using for and while

- We would like to store our IIEES waveform data in a tree like directory structure for the stations given in file `iiees.stn` in directory example4. The tree structure is station_name_year_month. Bash script `iiees_tree.bash` do the task. In this script we use `while` form of do loop in addition of the usual `for` command.

Example 4-5

Automation using external bash variables

- Here we would like to ask the range of years for which we create IIEES tree structure. There are two possibility for this. In the first method, we give the beginning and ending years as arguments to our bash program. This version is done in `script iiees_tree_1.bash`.

  In the second method, we use command read to interactively get the beginning and end years from user. This method is done in `script iiees_tree_2.bash`

A. Ghods aghods@iasbs.ac.ir

Example 4-6

Automation and if

- Here we would like to modify `iiees.tree_2.bash` so that if a given directory exists already, the script does not try to create it. We use `if` command in the script `iiees.tree_3.bash` for this task. `if` command can test if a file or directory exist or not.

A. Ghods aghods@iasbs.ac.ir

Hints for debugging

- Your bash script like any other program can have bugs. The best way to debug is to write step by step a large bash program and get proper output from different variables using command `echo`. If you have do loops, in order to make your program slow enough so to see the value of your variables, you can use command `sleep` to cause your program to pause for a given period of time at a specified spot on your script. You can also use `read`.

A. Ghods aghods@iasbs.ac.ir

Exercise 4

- Change the script for Example 4-3 to print the total size of files in kb for each directory and the sum for all directories.

A. Ghods aghods@iasbs.ac.ir

Section 5
Advanced examples

In this section, I will introduce some practical and rather advanced examples which I encountered in my own research. These examples uses all we have learned up to know in the previous slides.

Example 5-1

Extracting data from a big seismic data bank

- In example 4-4 we have made a big but empty data bank. The real data bank of IIEES has the same structure but filled with data. Put some files in one of the directory and write a bash script to find the file automatically by navigating the data bank!

A. Ghods aghods@iasbs.ac.ir

Example 5-2

Managing IRSC zip waveform files

- IRSC website (irsc.ut.ac.ir) gives waveforms for all events larger than magnitude 4. The name of zipped waveform files is composed of word "`wavout`" and some random numbers usually 3 to 5 digits. If we want to make our own organized data bank out of a set of disorganized irsc zip files, we need to write a bash script to arrange the file into a manageable data bank! This has been done in `examples_advanced/example_apsheron_irsc_datarsc_data_bank_maker`

A. Ghods aghods@iasbs.ac.ir

Example 5-3

Converting format of a large number of waveform files

- In this example, we would like to convert automatically SAC waveforms of all events in directory `Talesh_extracted_events` to nordic seisan format. The SAC data is from IASBS temporary network in north west of Iran. This task has been done in script `examples_advanced/Talesh_extracted_events/convert_all_sac_seisan_Talesh.bat`
- This script needs both seisan and SAC installed on your computer.

A. Ghods aghods@iasbs.ac.ir

Example 5-4

Running a given program with different inputs

- In this example, we will run VELEST program for 1-D inversion for crustal velocity using different initial velocity structure. This is actually a sensitivity analysis for stability of our inverted 1-D velocity model. This is done in file `bash_scripting/examples_advanced/VELEST6/script.bat`
- Rewrite the program using do loops !

A. Ghods aghods@iasbs.ac.ir

Example 5-5

Running programs using parallel strategy

- Now we would like to modify the previous script so that it runs in parallel. We should run each program in a different directory using `nohup` command. Write the program so that it asks for number of cpus of your computer. Do not forget to collect all of your output to the main directory!

End !

- Please let me know what you think about this presentation/workshop. I will be happy to have your questions. aghods@iasbs.ac.ir