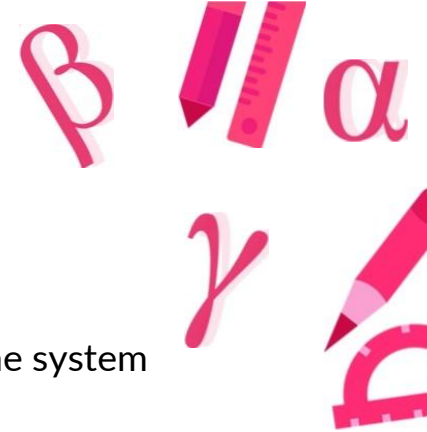
MODULE 7 :
Miscellaneous



Environment Variables - Basics

- Environment Variables are variables that store information about the system
- They can be used to store data, set configuration options and customize the shell environment under Linux
- Can be divided into two types:
  - System Environment Variables
  - Local variables

Environment Variables - Basics

System Variables:

- **Standard Names**
  - Used by the Shell
  - Normally they are All Caps
  - More can be added by the users for their usage

- **Local Variables**
  - User selected names
  - Local to a shell (not passed to children shells or programs)
  - Convention is to avoid all caps to differentiate them

Environment Variables - Usage

- Examples of use of Environment Variables (not a full list) :
  - Configure look and feel of shell such as colors and bash prompt
  - Time zone, host name,…
  - Search path for executables, or any types of files
  - Default values for some system configurations
  - Some configuration options for specific programs

Process Environment

- Linux does not maintain or store a global set of environment variable for the system
- Each running program (process) will have its own environment settings
- This means different processes may have different environment settings
- The environment settings for each running process in the system can be listed by viewing the file /proc//environ
    - Where pid is the Process ID

Process Environment

Process receive their environments settings by:

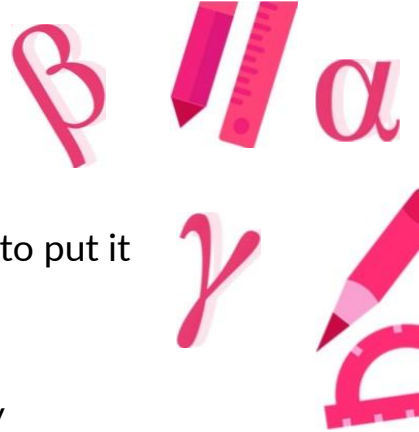
- **By inheritance**
  - Each process will have a parent process that started it
  - The child process inherits the environment settings of its parent process
  - that each program (process) that is started inside the shell, is a child of that shell, hence processes started from the shell, inherit the shell environment settings
  - a non-login shell is a child of a login shell, hence it inherits its environment settings at startup
  - local variables are not inherited to child shells or processes

Process Environment

Process receive their environments settings by:

- **By Startup Scripts**
  - Some programs source some scripts at startup
  - These scripts may include some environment settings that is added to the process settings inherited from its parent
  - We have already discussed this for login/non-login shell startup
  - Login Shells /etc/profile ~/.bash-profile or ~/.bash-login or ~/.profile
  - Non-Login Shells /etc/.bashrc or /etc/bash.bashrc ~/.bashrc
  - GUI Applications (applications started from the GUI) ~/.xinitrc

/etc/profile



- To add settings that will apply to all shells, and all users… we need to put it in /etc/profile
- In most distributions, it is preferred not to edit /etc/profile directly
- To enable that, /etc/profile has a loop that sources all scripts with extentions *.sh in the folder /etc/profile.d
- Accordingly, all we need to do is to put our settings in a new script file inside this folder and call it something.sh then make it executable
- Our script will be called from /etc/profile and hence our settings will be read by login shells, and inherited by non-login shells

export

- To Set a local variable in the shell $ My_Var=5 This way My_Var will not be inherited to any child or process of the current shell
- To Convert it into an Environment Variable $ export My_Var This way My_Var will be inherited to any child shell or process of current shell
- To bring it back to be just a local variable $ export -n My_Var
- To reset an Environment variable $ export My_Var=
- To Completely remove the variable $ unset My_Var

List Environment variables

- set
- Printenv
- env

Common Environment variables: PATH

- It is a list of directories separated by a colon ":" /home/tom/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
- This list represents the search path for commands and binaries, when you issue a command
- To show the current search path $ echo $PATH
- To add a folder to the end of the path $ export PATH=$PATH:/usr/bin
- To add a folder to the beginning of the path $ export PATH=/bin:${PATH}

Common Environment variables: PS1

- Responsible for setting the shell prompt
  - \u →username
  - \h →hostname
  - \W →current working directory

- Example $ export PS1=[\u@\h \W]\$

Common Environment variables: SHELL

- Contains the path to the login shell

- Example:
  - $ echo $SHELL /bin/bash

Common Environment Variables:

- EDITOR
- TERM
- HOME
- HOSTNAME

Links in Unix

- On the UNIX command line, the tool ln abbreviates the term link.
- It allows you to create an additional reference to a file, or directory.
- It does that by adding an additional name of an entry in the file allocation table of the file system.
- It is an essential tool in unix and it is part of the package coreutils.

```
_ dpkg -S `which ln`
coreutils: /bin/ln
```

Links in Unix(contd.)

Links on unix file system are of two types:

Hardlink

- Linking files by reference
- System maintains a count of the number of links
- Does not work across file system

Softlink or Symbolic link

- Linking files by name
- No counter is maintained
- Work across file system

```
root in /home/akhan/Documents/demo/important
_ ls
file1.txt  file2.txt  file3.txt  file4.txt

root in /home/akhan/Documents/demo/important
_ stat file1.txt
  File: file1.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 803h/2051d      Inode: 9700532     Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2019-10-15 17:09:20.130714453 +0530
Modify: 2019-10-15 17:09:20.130714453 +0530
Change: 2019-10-16 12:07:14.200475126 +0530
 Birth: -

root in /home/akhan/Documents/demo/important
_ ln file1.txt file5.txt

root in /home/akhan/Documents/demo/important
_ ls -ai
9700524 .   9700532 file1.txt  9700534 file3.txt  9700532 file5.txt
9699798 ..  9700533 file2.txt  9700535 file4.txt

root in /home/akhan/Documents/demo/important
_ stat file1.txt
  File: file1.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 803h/2051d      Inode: 9700532     Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2019-10-15 17:09:20.130714453 +0530
Modify: 2019-10-15 17:09:20.130714453 +0530
Change: 2019-10-16 12:07:32.532663746 +0530
 Birth: -
```

A hard link is a UNIX path name for a file.

It is created with `ln` command.

```
root in /home/akhan/Documents/demo/important
_ stat file1.txt
  File: file1.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 803h/2051d      Inode: 9700532     Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2019-10-15 17:09:20.130714453 +0530
Modify: 2019-10-15 17:09:20.130714453 +0530
Change: 2019-10-16 12:07:32.532663746 +0530
 Birth: -

root in /home/akhan/Documents/demo/important
_ ln -s file1.txt shortcut_file1

root in /home/akhan/Documents/demo/important
_ ls -ai
9700524 .   9700532 file1.txt  9700534 file3.txt  9700532 file5.txt
9699798 ..  9700533 file2.txt  9700535 file4.txt  9699601 shortcut_file1

root in /home/akhan/Documents/demo/important
_ stat file1.txt
  File: file1.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 803h/2051d      Inode: 9700532     Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2019-10-15 17:09:20.130714453 +0530
Modify: 2019-10-15 17:09:20.130714453 +0530
Change: 2019-10-16 12:07:32.532663746 +0530
 Birth: -

root in /home/akhan/Documents/demo/important
_ ls -l shortcut_file1
lrwxrwxrwx 1 root root 9 Oct 16 12:22 shortcut_file1 -> file1.txt
```

Symbolic link is also a means of referencing a file.

It is created with ln -s command.

Key Takeaways on Links

Hardlink	Softlink
Does not create a new inode	Create a new inode
Cannot link directories unless it is done by root	Can link directories
Cannot link files across file systems	Can link files across file system
Increase hard link count of the linked inode	Does not change hard link count of the linked inode

Thank You!