UNIT 10: PROCESS

UNIT STRUCTURE

10.1 Learning Objectives

10.2 Introduction

10.2.1 Types of Process

10.2.2 Starting a Process

10.2.3 Listing Running Processes

10.2.4 Stopping a Process

10.2.5 Attributes of a Process

10.2.6 Job, Task and Process

10.3 Job Control

10.4 Crontab File Format

10.5 Linux Backup and Recovery

10.6 Let Us Sum Up

10.7 Further Readings

10.8 Answer to Check Your Progress

10.9 Model Questions

10.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- understand about services and processes in Linux
- learn the basic commands for process management
- explain the attributes of process status word
- describe job, task and processes
- understand crontab file format and its use
- understand to backup Linux system and restore from backup.

10.2 INTRODUCTION

In the previous unit, we have learnt the concept of shell script. We have discussed the use of conditional and loop statements in a shell script. We have also discussed how functions can be used in a shell script. In this

unit, we will discuss the various tasks associated with a process like starting, listing, and stopping a process.

Process: A program in running state (execution) is called a process. When a program is executed on Unix/ Linux operating system, the operating system creates an environment for the program. This environment contains all the resources required by the program to execute in the system if no other processes were running on the system.

When we run a command in Unix/ Linux operating system, it creates or starts a new process. For example, when we execute the ls command, a process is started to list all the directory and files in the current directory and display it. A process, in simple terms, is an instance of a running program.

When a process starts executing, the operating system assigns a process ID (pid) to the process. This process ID is unique for all the processes running in the system but after termination of the process, same process id may be assigned to some other process. Linux uses PID to track each process in the system.

Service: Linux systems provide variety of system services (process management, login, syslog, cron, etc.), network services (remote login, e-mail, printers, web hosting, data storage, file transfer, domain name resolution, dynamic IP address assignment) and much more.

A service is a program or an application in Linux that runs in the background. It runs without the need of user intervention and sometimes users are unaware of it. Most of the services stated by the operating system for smooth running of the system. Generally, a Linux service has the following characteristics:

- A service does not have a graphical interface. So, users cannot interact with services through graphical interface.
- Most of the services are started with the operating system. Third-party services can be configured to start or not with the system by the user. Even few third-party services can be started by the user as per requirement.

- Services running in the background, wait for certain signal to start the processing. For example, mail demon checks for new email and displays it when any new mail is received.

Linux supports different ways to manage services such as start, stop, restart, enable auto-start at system boot, etc.

10.2.1 Types of Process

Process in Linux can be of two types-

- Foreground Processes
- Background Processes

Foreground Processes: By default, every process that we start runs in the foreground. A foreground process reads input from standard input device and displays the output in the standard output device. For example, if we run ls command, it will list all the files in the current directory on the terminal. Most of the foreground processes can be run using graphical user interface, for example media player, word processing software etc.

When a process runs in the foreground, no other process can run (or start any other processes) even if the process is time-consuming. The terminal (or command prompt) become inaccessible when a process runs in the foreground until the process finishes its execution.

Background Processes: A background process runs in the background without being directly connected to the input/output devices. When a background process requires any input or need to display an output, it waits for the system to make them available. Most of the background process are stated by the system for smooth functioning of the system. The advantage of running a background process is that we can run other processes in the system at the same time; we need not have to wait for completion of the background process to start another process.

If a process running in the background needs any input, it

goes into waiting state until some other process moves it into the foreground and provides the data from standard input device.

10.2.2 Starting a process

A process can be run either in foreground or in background.

Foreground Process: Every process when started runs in foreground by default, receives input from the keyboard, and sends output to the screen. Foreground processes can be run from graphical user interface by typing command in the terminal. For example

$ pwd

will give the current working directory of the user like

/home/student

Background Process: Background process can be started by adding '&' along with the command.

$ pwd &

will give the following output

[1] 2567

/home/student

Since pwd does not require any input from the keyboard, it goes to the stop state until moved to the foreground.

10.2.3 Listing Running Processes

We can list the running processes started by current user by using the ps (process status) command as follows ?

$ps

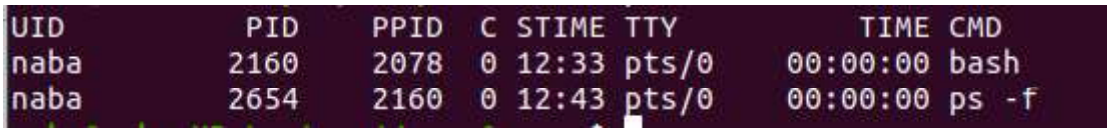
OUTPUT:

```
  PID TTY          TIME CMD
 2160 pts/0    00:00:00 bash
 2583 pts/0    00:00:00 ps
```

One of the most commonly used flags for ps is the -f (f for full) option, this provides more information about the currently running processes ?

$ps -f

OUTPUT:



Description of the fields displayed by ps -f command is given below:

Sl. No.	Column	Description
1	**UID**	User ID that this process belongs to (the person running it)
2	**PID**	Process ID
3	**PPID**	Parent process ID (the ID of the process that started it)
4	**C**	CPU utilization of process
5	**STIME**	Process start time
6	**TTY**	Terminal type associated with the process
7	**TIME**	CPU time taken by the process
8	**CMD**	The command that started this process

Other common options used with ps command are -

Sl. No.	Option	Description
1	-a	Displays all processes on a terminal, with the exception of group leaders.
2	-c	Displays scheduler data.
3	-d	Displays all processes with the exception of session leaders.
4	-e	Displays all processes.
5	-f	Displays a full listing.
6	-g *list*	Displays data for the *list* of group leader IDs.
7	-j	Displays the process group ID and session ID.
8	-l	Displays a long listing
9	-p *list*	Displays data for the *list* of process IDs.
10	-s *list*	Displays data for the *list* of session leader IDs.
11	-t *list*	Displays data for the *list* of terminals.
12	-u *list*	Displays data for the *list* of usernames.

10.2.4 Stopping a Process

Process can be stopped by several different ways. If the process is running on console, we can stop that process by pressing CTRL + C. In Linux, CTRL + C is used to send a SIGINT signal, which cancels or terminates the currently-running program. This works when the process is running in the foreground mode.

If we want to run a process which is not currently running on the console or the process is running in the background, then by

using ps command we can get the process ID. The kill command is used to kill the process by using process ID. Following example shows the use of kill command ?

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2160    2078  0 12:33 pts/0    00:00:00 bash
naba        2988    2160 10 12:52 pts/0    00:00:00 gedit
naba        2995    2160  0 12:52 pts/0    00:00:00 ps -f
naba@naba-HP-Laptop-14q-cs0xxx:~$ kill 2988
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2160    2078  0 12:33 pts/0    00:00:00 bash
naba        3001    2160  0 12:52 pts/0    00:00:00 ps -f
[1]+  Terminated              gedit
```

Here, the kill command terminates the process gedit (PID 2298). If a process cannot be killed by kill command, then kill -9 command is used to kill the process by force. Following example shows the use of kill -9 command :

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2847    2078  0 12:48 pts/2    00:00:00 bash
naba        2855    2847  2 12:48 pts/2    00:00:00 gedit
naba        2864    2847  0 12:49 pts/2    00:00:00 ps -f
naba@naba-HP-Laptop-14q-cs0xxx:~$ kill 2855
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2847    2078  0 12:48 pts/2    00:00:00 bash
naba        2855    2847  1 12:48 pts/2    00:00:00 gedit
naba        2932    2847  0 12:49 pts/2    00:00:00 ps -f
naba@naba-HP-Laptop-14q-cs0xxx:~$ kill -9 2855
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2847    2078  0 12:48 pts/2    00:00:00 bash
naba        2939    2847  0 12:50 pts/2    00:00:00 ps -f
[1]+  Killed                  gedit
```

Parent and Child Processes

All the processes in Linux have two ID numbers assigned to it- Process ID (PID) and Parent process ID (PPID). Each process in the system has a parent process. Parent process is the process which starts the current process.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
naba        2847    2078  0 12:48 pts/2    00:00:00 bash
naba        2855    2847  2 12:48 pts/2    00:00:00 gedit
naba        2864    2847  0 12:49 pts/2    00:00:00 ps -f
```

In the above example the terminal process id (bash) is 2847 and the other two processes gedit and ps-f are executed in the terminal, so parent process of both the processes gedit and ps-f is bash(terminal) which is shown as above. Most of the commands that we run have the shell as their parent process as we execute those commands from the shell.

The top Command

The top (table of processes) command shows a real-time view of running processes in Linux and displays kernel-managed tasks. The command also provides a system information summary that shows resource utilization, including CPU and memory usage. The top command is a very useful tool for quickly displaying the currently running processes sorted by various criteria. It is an interactive command and updates frequently by showing information about physical and virtual memory, CPU usage, load averages, and busy processes. Following example shows the top command and its output ?

$top

```
top - 12:57:12 up 24 min,  1 user,  load average: 0.32, 0.35, 0.29
Tasks: 227 total,   1 running, 226 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.8 us,  1.3 sy,  1.8 ni, 93.8 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0 st
MiB Mem :   7820.6 total,   4625.5 free,   1162.7 used,   2032.4 buff/cache
MiB Swap:   3906.0 total,   3906.0 free,      0.0 used.   6185.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1155 naba      20   0 4180088 253204 104836 S  11.3   3.2   1:10.49 gnome-shell
   2357 _apt      25   5   24056   9368   8340 S   5.0   0.1   0:21.10 http
    856 naba      20   0  570140  88504  51956 S   4.3   1.1   1:16.89 Xorg
   2272 naba      30  10  946720 174336  99796 S   2.3   2.2   0:38.98 update-manager
    414 root     -51   0       0      0      0 S   0.3   0.0   0:19.36 irq/135-rtw88_p
    646 message+  20   0    9828   6232   3812 S   0.3   0.1   0:05.68 dbus-daemon
   2703 root      0 -20       0      0      0 I   0.3   0.0   0:00.53 kworker/u9:1-i915_flip
   3055 root     20   0       0      0      0 I   0.3   0.0   0:00.36 kworker/1:2-events
   3071 root      0 -20       0      0      0 I   0.3   0.0   0:00.11 kworker/u9:2-rtw_tx_wq
   3080 naba     20   0   12000   3872   3216 R   0.3   0.0   0:00.12 top
      1 root     20   0  167616  11552   8352 S   0.0   0.1   0:01.40 systemd
      2 root     20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root      0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root      0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      6 root      0 -20       0      0      0 I   0.0   0.0   0:00.11 kworker/0:0H-acpi_thermal_pm
      8 root     20   0       0      0      0 I   0.0   0.0   0:00.47 kworker/u8:0-i915
      9 root      0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     10 root     20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rude_
     11 root     20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_trace
     12 root     20   0       0      0      0 S   0.0   0.0   0:00.07 ksoftirqd/0
     13 root     20   0       0      0      0 I   0.0   0.0   0:01.04 rcu_sched
     14 root     rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/0
```

10.2.5 Attributes of a process

When a program starts executing, it is called a process. When a process starts, some attributes are set to the process. These attributes are assigned by the Operating System to create the process control block (PCB) for each process. Attributes of the process include the following.

Process ID: When a process is created, a unique id is assigned to the process called the process ID or PID. PID is used to uniquely identify the process in the system.

Program counter: A program counter is a register, which contains the address (location) of the instruction being executed at the current time. As each instruction gets fetched, the program counter increases its value by 1. After each instruction is fetched, the program counter points to the next instruction to be executed. When a process is suspended, the program counter contains the instruction number to be executed after resumption of the process. Process State: The Process, from its creation to the completion, goes through various states which are new, ready, running and waiting. Process switches from one state to another depending on various condition. When a program is ready for execution for the first time it enters to new state, once job scheduler selects the process for execution it enters to the ready state. When processor choose the process for execution, it enters to the running state. Whenever process needs any I/O, it enters to the waiting state (waiting for I/O). If process swich occurs or processor in the waiting state completes its I/O, then again it enters to the ready state. When process completes its task, it enters to the terminate state. The process

transition diagram is depicted in figure 1.

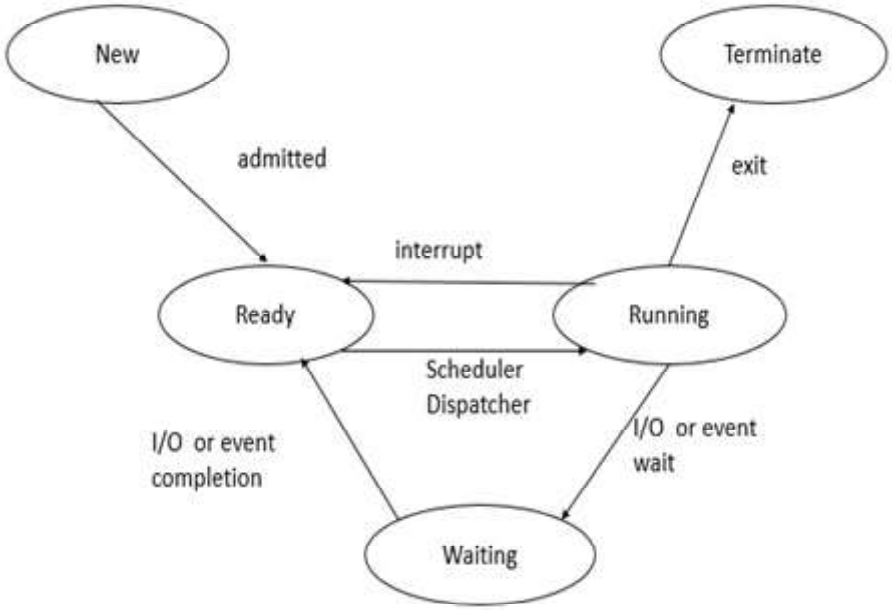


Figure 1. Process transition diagram

Priority: Every process has its own priority and this priority is set by the operating system (OS). The process with the highest priority gets the CPU first. This information is also stored on the process control block.

General Purpose Registers: Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.

List of open files: During the execution, every process uses some files which need to be present in the main memory. Process control block stores the information about the open files. All the files opened by the process are closed when process is terminated.

List of open devices: The devices used by the process is maintained in the process control block. OS also maintains the list of all open devices which are used during the execution of the process.

10.2.6 Job, Task and Process

Job: A job is a complete unit of work under execution. A job consists

of many tasks which consist of many processes. A job is a series of tasks in a batch mode. Programs are written to execute a job. Jobs and tasks are used synonymously in computational work.

Task: Task is a unit of work being executed. Task in Operating System may be synonymous with process. A task is a subpart of a job. Tasks combine to form a job. The task may be a thread, process, a single job and much more.

Process: The process is a series of actions that is done for a particular purpose.

Job and task define the work to be done, whereas process defines the way the work can be done or how the work should be done.

The concept of job, process and task revolves around each other. Job, task and process may be considered the same or different in reference to the context they refer to. A process is an isolated entity of Operating System. A task may be called a process if it is a single task. A job may be called a task if the job to be performed is a single unit of work. A process or group of processes can be termed as a task and a group of tasks can be termed as a job.

10.3 JOB CONTROL

In Unix and Unix-like operating systems, job control refers to control of jobs in the shell. Basic job control features are suspending, resuming, or terminating of all processes in the job or process group. Job control is of particular interest in Unix due to its multiprocessing nature, and it is different from job control in process management.

Typically, the shell keeps a list of jobs in a job table. The jobs command lists the background jobs existing in the job table, along with their job number and job state (stopped or running). When the user exits from the shell, the shell sends SIGHUP to all jobs, and waits for the process groups to end before terminating itself.

The 'disown' command is used to remove link from the parent process and make the processes orphan, such that when the session ends,

shell should not wait for them to terminate nor shell required to send SIGHUP to those processes.

A job running in the foreground can be stopped by pressing the suspend command (Ctrl + Z). This sends the "terminal stop" signal (SIGTSTP) to the process group. By default, SIGTSTP causes processes receiving it to stop, and control is returned to the shell. A process can also be paused with the "stop" signal (SIGSTOP).

A job running in the foreground can be interrupted by typing the interruption command (Ctrl + C). This sends the "interrupt" signal (SIGINT), which defaults to terminating the process.

A stopped job can be resumed as a background job with the 'bg' command, or as the foreground job with 'fg' command. In either case, the shell redirects I/O appropriately, and sends the SIGCONT signal to the process, which causes the operating system to resume its execution.

A background process that attempts to read or write its sent terminal SIGTTIN (for input) or SIGTTOU (for output) signal. Shells often override the default stop action of SIGTTOU so that background processes deliver their output to the controlling terminal by default.

bg: Resume a suspended program without bringing it to the foreground.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs
[1]   Running                 gedit &
[2]-  Stopped                 vi
[4]+  Stopped                 vi
naba@naba-HP-Laptop-14q-cs0xxx:~$ bg %2
[2]- vi &

[2]+  Stopped                 vi
```

fg: Resumes a suspended job and brings it to the foreground.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs
[1]   Running                 gedit &
[2]+  Stopped                 vi
[4]-  Stopped                 vi
naba@naba-HP-Laptop-14q-cs0xxx:~$ fg %1
gedit
```

jobs: List the status of all running jobs

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs
[1]-  Running                 gedit &
[2]+  Stopped                 vi
```

Jobs command has following options

-l Lists process IDs in addition to the normal information.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -l
[1]-  2274 Running                 gedit &
[2]+  2352 Stopped (tty output)    vi
```

-n Displays information only about jobs that have changed status since the user was last notified of their status.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -n
[4]+  Stopped                 vi
```

-p Lists only the process ID of the job's process group leader.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -p
2274
2352
2664

[3]+  Stopped                 vi
```

-r Displays only running jobs.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -r
[1]   Running                 gedit &
```

-s Displays only stopped jobs.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ jobs -s
[2]   Stopped                 vi
[3]-  Stopped                 vi
[4]+  Stopped                 vi
```

kill: Send a signal to a process, affecting its behaviour or killing it.

wait- It waits for the process to change its state i.e. it waits for any running process to complete and returns the exit status.

```
naba@naba-HP-Laptop-14q-cs0xxx:~$ wait
bash: wait: warning: job 2[2352] stopped
bash: wait: warning: job 3[2664] stopped
bash: wait: warning: job 4[2684] stopped
```

suspend: Suspends the execution of this shell until it receives a SIGCONT signal.

10.4 CRONTAB FILE FORMAT

Cron is named after Greek word "Chronos" meant for time. Cron is a system process that will automatically perform tasks as per specific schedule. It is a set of commands that are used to run as per schedule. Crontab stands for "cron table". It allows to use job scheduler to execute tasks. Cron is a system daemon used to execute desired tasks (in the background) at designated times.

A crontab file is a simple text file containing a list of commands intended to be run at specified times. It is edited using the crontab command. The commands in the crontab file (and their run times) are checked by the cron daemon, which executes them in the background.

Each user (including root) has a crontab file. The cron daemon checks a user's crontab file regardless of whether the user is actually logged into the system or not.

Linux Crontab format

Crontab of Linux has six fields. The first five fields define the time and date of execution, and the 6th field is used for command execution.

Crontab syntax:

[MIN] [HOUR] [DOM] [MON] [DOW] [CMD]

For example

30 08 10 06 * gedit

31 06 * * * ps

01 * * * * jobs

Here, asterics (*) is used for matching, we can use range of values (for eg- 1-10 or 30-40 or jan-mar, mon-wed) or multiple ranges (apr-jun,oct-dec) in place of values. The values in each field are from the valid range of values.

Crontab file allowed ranges

Field	Description	Allowed Value
MIN	Minute field	0 to 59
HOUR	Hour field	0 to 23
DOM	Day of Month	1-31

MON	Month field	1-12
DOW	Day Of Week	0-6
CMD	Command	Any command to be executed.

Cron special keywords and its meaning

Keyword	**Equivalent**
@yearly	0 0 1 1 *
@daily	0 0 * * *
@hourly	0 * * * *
@reboot	Run at startup.

These special keywords are useful when we need to run a particular command after a definite interval.

Use of cronjobs

Reasons for using Cronjobs in Linux are considered as following-

- It helps OS to take a scheduled backup of log files or database.
- It deletes old log files.
- It archives and purges database tables.
- Send out any notification email such as Newsletters, Password expiration email.
- Regular clean-up of cached data.
- Crontab is an ideal option to automate Unix jobs.
- It is used to automate system maintenance.

10.5 LINUX BACKUP AND RECOVERY

In Linux, backup can be made using various ways such as tar, dd, SCP, etc. Tar, also known as tape archiving, is the most commonly used to create backup and restore. We can compress and archive files and directories using tar. Furthermore, we can use the tar command to create a single backup file for more than one file. The syntax of the tar command is:

tar <options> <file>

where we can use many options, which are discussed below:

c = creates a new archive

v = provides verbose output

f = specifies the archive file to use

t = lists the files in an archive

x = Extract the backup

z = Zipping

For example, if we want to create a backup file (say abc.txt) using tar, and if the backup file is to be compressed, then we can use the following command:

#tar -czvf abc.tar.gz abc.txt

If we want the system to run the backup automatically, we can use the crontab file. The crontab file contains a list of commands that can be run regularly. Every line in the crontab file is for performing a task at a particular time. By using this file, we can automate our task or schedule a task to perform at a particular time. If we want to take backup at 6 AM every morning then we need to add the following in the users crontab file.

* 6 * * * tar -czvf abc.tar.gz abc.txt

Incremental Backups with Tar:

Incremental backup is the advanced backup type using tar. Here, we do not have to perform the full back up every time. We need to back up only the data that has been created or modified since the last backup has been taken. So, using this type of backup, we can save backup time and disk space. Steps to perform the incremental backup is as follows:

At first, we have to perform the full backup. Let us consider an example where we will perform the full backup of directory data. This can be done using the following way:

#tar --verbose --verbose --create --gzip --listed-incremental=/backup/data.sngz --file=/backup/data.tgz data

where the option /backup/data.tgz will compress the data directory in tgz format and save it inside /backup directory and /backup/data.sngz option will create an initial incremental backup (or full backup) and save it at /backup/data.sngz.

After a few days and after modifying the data directory, if we want to perform the incremental backup in the data directory, we will use the following command:

#tar --verbose --verbose --create --gzip --listed-incremental=/backup/data.sngz --file=/backup/data1.tgz data

Restore the Backup with Tar Incremental Backup

To restore the incremental backup, we can use the following command:

tar --extract --listed-incremental=/dev/null --file data.tgz

Creating backup with Timeshift application

For creating system backup on Ubuntu 20.04 system, we need to install Timeshift backup utility as follows-

$ sudo apt install timeshift

Timeshift is a graphical user interface for creating backup and restore point on ubuntu 20.04.

Create Backup by using the Timeshift GUI

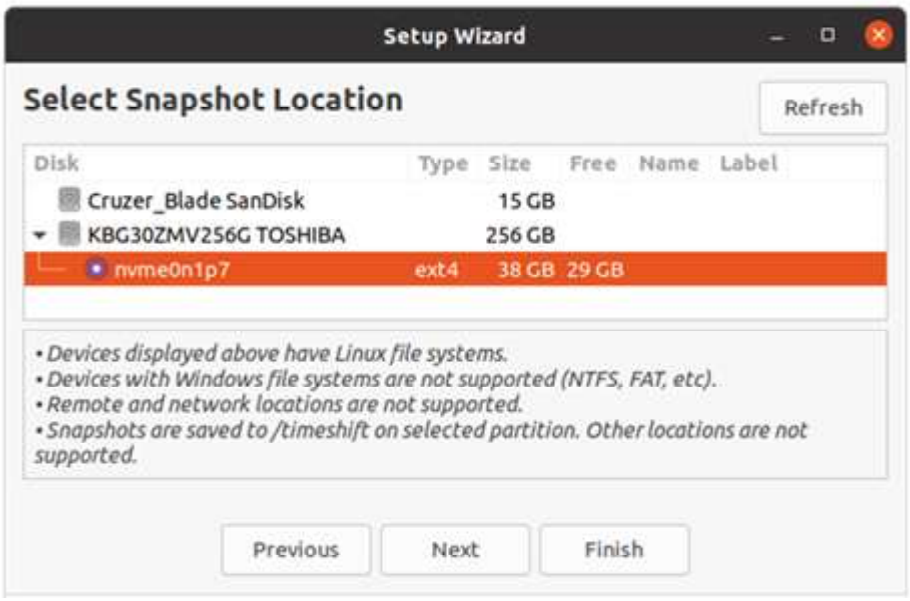
To create system backup

Step1: First to Open the timeshift application from Activities menu.



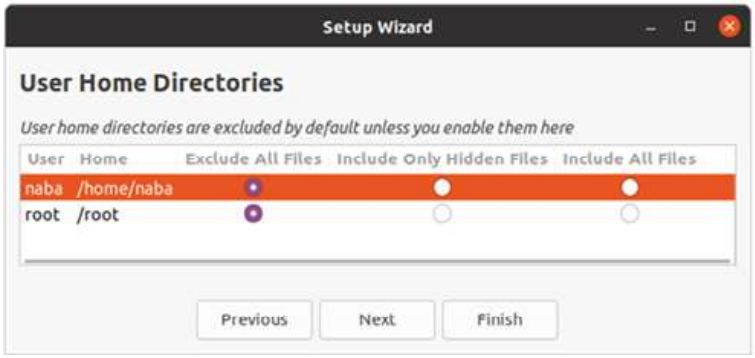
Upon opening the timeshift application user will be greeted with a wizard to help schedule the backups. Here there are two options. First is to use the rsync as a main backup tool or take an advantage of an inherent BRTFS built-in file system features. In this example, we will use rsync. Select RSYNC and hit the Next button.

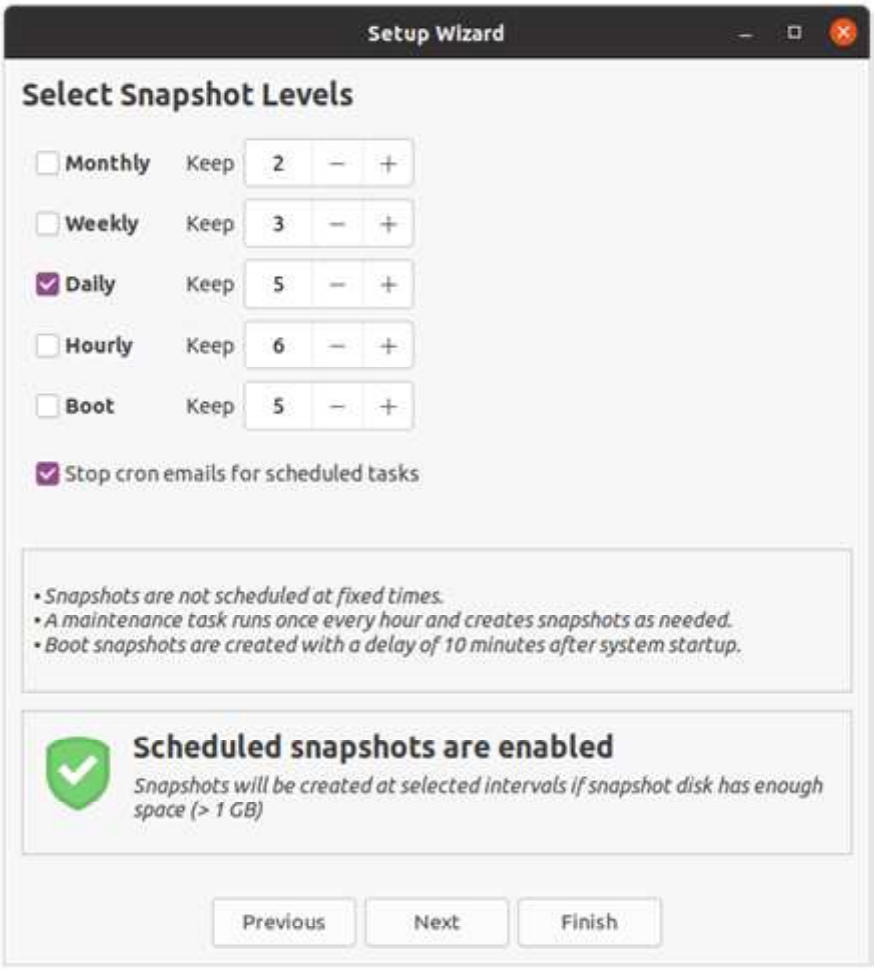
Step 2: Then, we have to select backup destination. timeshift will search our system for available file-system partition and provide us with an option to where to create backup file.



Step 3: Then to select how often we wish to perform the system backup and how many backup snapshots will be retrained before the first backup is overwritten.

Step 4: As the screenshot indicates the home directories are excluded by default. Depending on our work environment, we have to select whether we wish to include home directories into the backup.



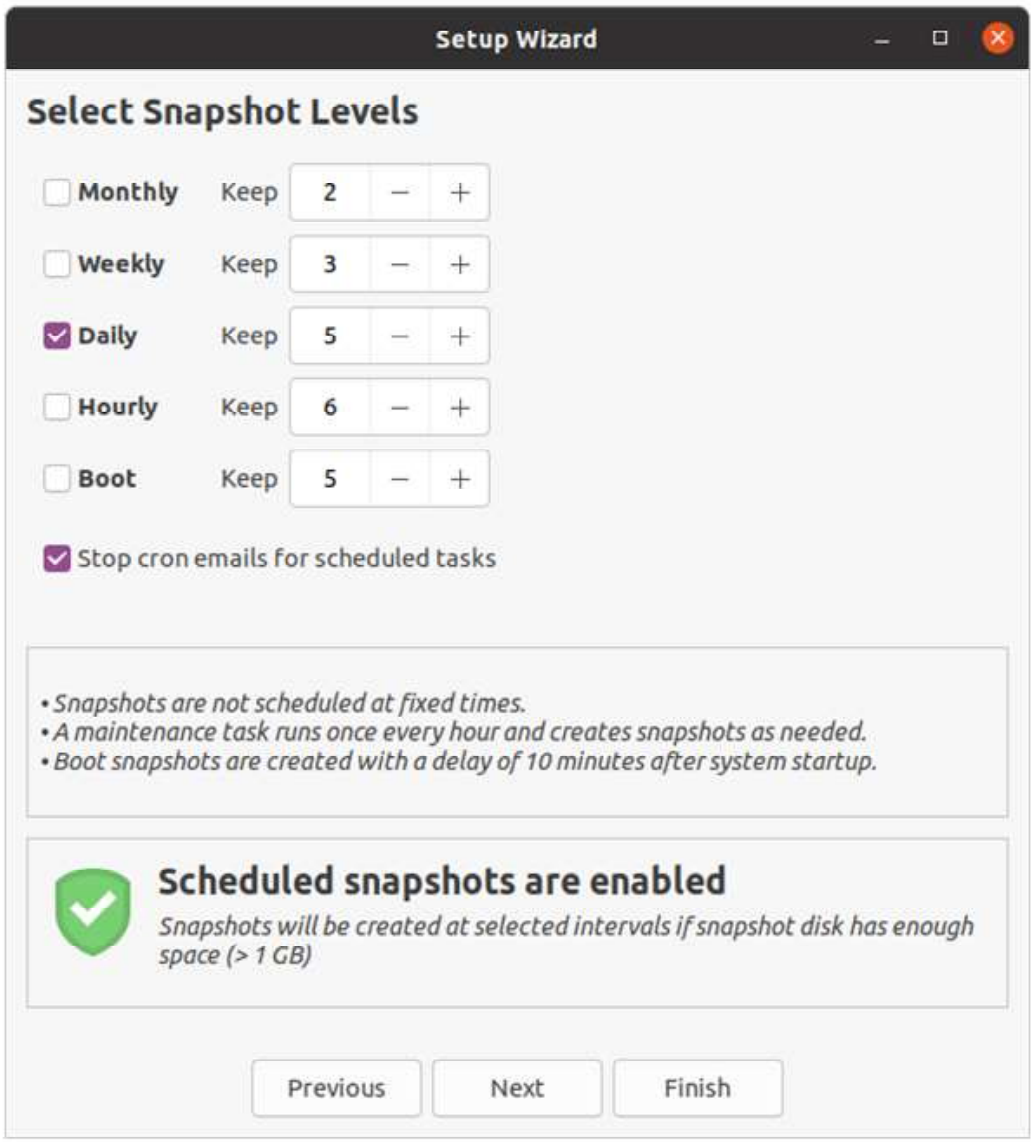


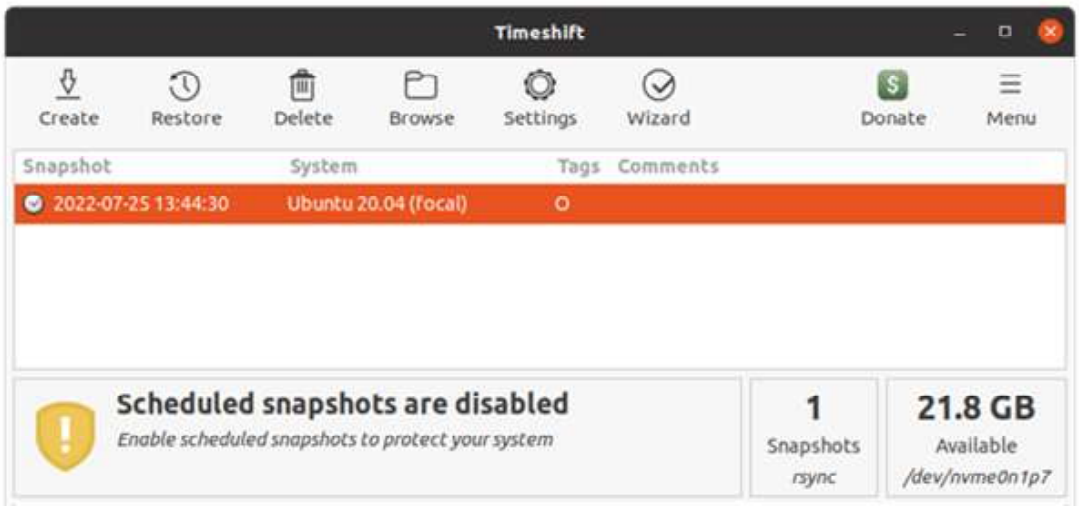
Step 5: This will conclude our initial backup schedule setup. We have to click on Finish button.

Step 6: The backup has not been created yet. We can either wait until the timeshift automatically triggers the backup or simply hit the Create button to perform the previously predefined backup now.

Step 7: We have to wait for the backup to complete.

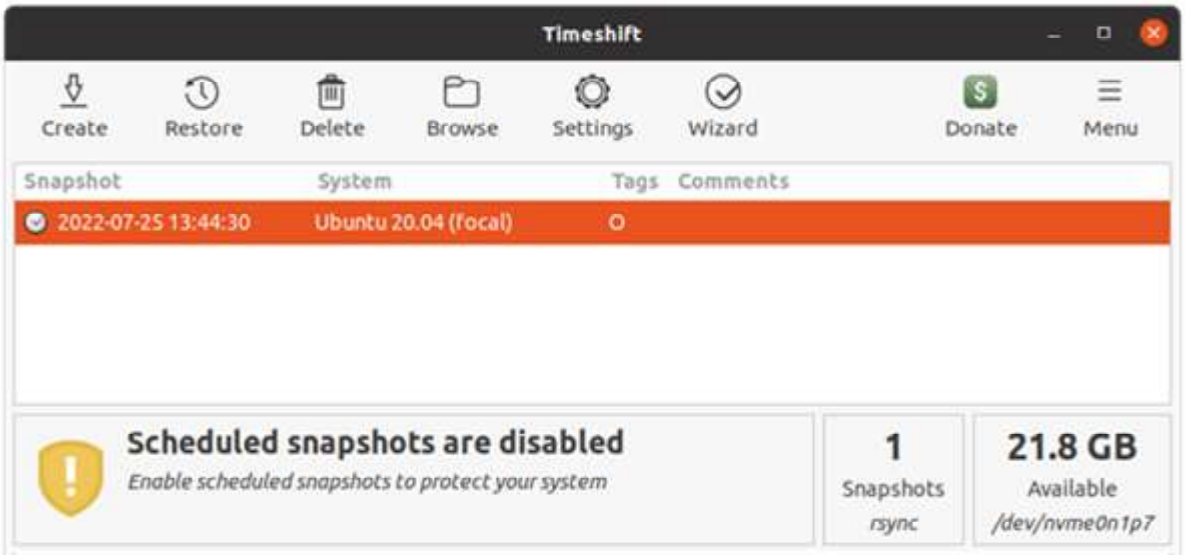
Step 8: If everything is done well, we should see our first backup snapshot listed.


Setup Wizard
Select Snapshot Levels
Monthly Keep 2
Weekly Keep 3
Daily Keep 5
Hourly Keep 6
Boot Keep 5
Stop cron emails for scheduled tasks
• Snapshots are not scheduled at fixed times.
• A maintenance task runs once every hour and creates snapshots as needed.
• Boot snapshots are created with a delay of 10 minutes after system startup.
Scheduled snapshots are enabled
Snapshots will be created at selected intervals if snapshot disk has enough space (> 1 GB)
Previous
Next
Finish


Timeshift
Create
Restore
Delete
Browse
Settings
Wizard
Donate
Menu
Snapshot
System
Tags
Comments
2022-07-25 13:44:30
Ubuntu 20.04 (focal)
O
Scheduled snapshots are disabled
Enable scheduled snapshots to protect your system
1
Snapshots
rsync
21.8 GB
Available
/dev/nvme0n1p7

Restore from Backup

Step 1: At this point we will restore the system from the previously created system backup snapshot. Select a backup snapshot from which we wish to restore and click on the Restore button.



Step2: Timeshift give us option on how to restore from the backup.



Step 3: The Timeshit will provide us with a list of changes it will take to restore from the backup just to make sure no data is lost in the process.

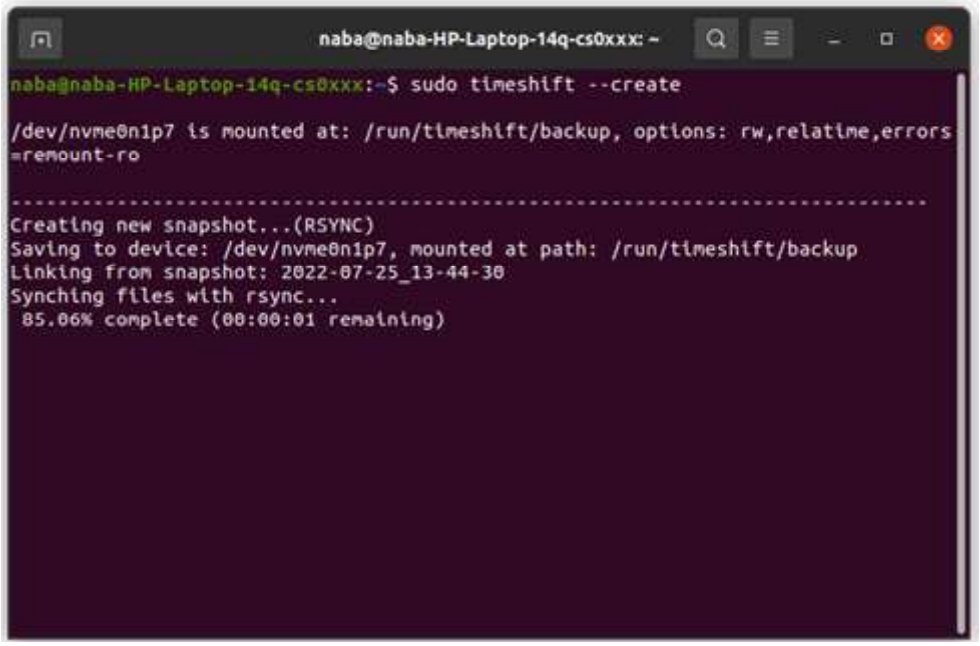
Step 4: Once we hit the Next button, the system will be restored

and restarted. All done.

Create and restore backup by using the Timeshift's command line

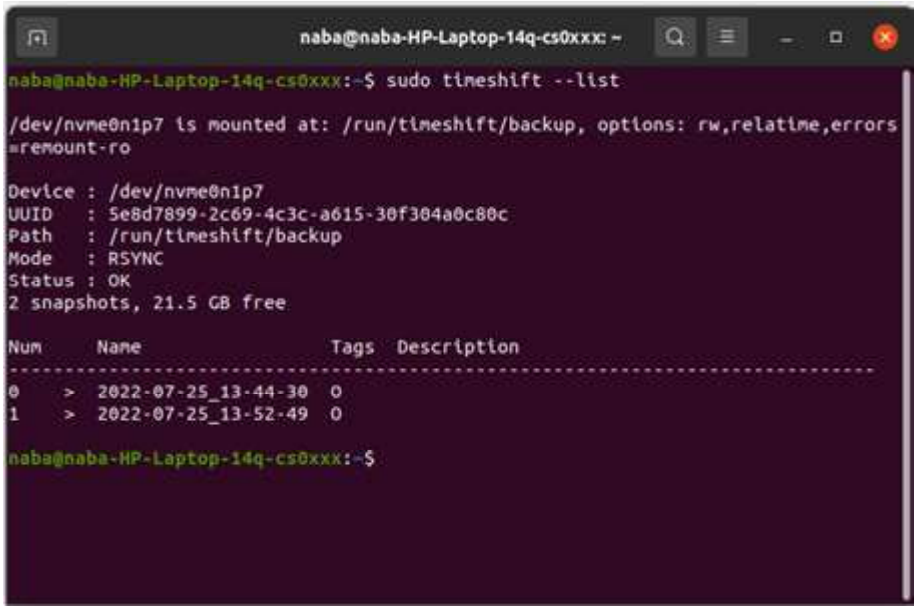
To create backup using command line, we need to execute the following command:

$ sudo timeshift --create



To List all the currently created system backup, following command is used which prints as follows:

$ sudo timeshift -list



Restore from the backup snapshot:

$ sudo timeshift --restore --snapshot "2022-07-25_13-52-49"

Here "2022-07-25_13-52-49" is the name of the backup file

Delete selected backup snapshot:

$ sudo timeshift --delete --snapshot "2022-07-25_13-52-49"



CHECK YOUR PROGRESS

Q 1: Which of the following command is used to make a background process to foreground?

GB / background / fg / bgIn

Q 2: How to run a process in the background of Linux process management?

Q 3: Process information in the current shell can be obtained by using

(gf / bg / gb / ps)

Q 4: A job is a set of task. (True / False)

Q 5: To stop a process by force, which command is used?

Q 6: Cron is named after Greek word "Chronos" means _____.

Q 7: PPID stands for _____.



10.6 LET US SUM UP

- A program in running state (execution) is called a process.
- When a program is executed in system, the system creates an environment for the program. This environment contains all the resources required by the program to ex-ecute.
- When a process starts executing, the operating system assigns a process ID (pid) to the process.
- A service is a program or application in Linux that runs in the background. A service runs without the need of user intervention and sometimes users are unaware of it.
- We can list the running processes started by current user by using the ps command.

- The top (table of processes) command shows a real-time view of running processes in Linux.
- Attributes of a process are: Process ID, Program counter, Process State, Priority, General Purpose Registers, List of open files, List of open devices.
- A job is a complete unit of work under execution. A job consists of many tasks which consist of many processes.
- In Linux, backup can be made using various ways such as tar, dd, SCP, etc.
- Backup can also be performed using Timeshift application.

10.7 FURTHER READINGS

1) Shell Scripting by Jason Cannon
2) Linux Command Line and Shell Scripting Bible by Richard Blum & Christine Bresnahan.
3) Mastering Linux Shell Scripting by Andrew Mallett & Mokhtar Ebrahim

10.8 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1: fg

Ans to Q No 2: by adding & after the command

Ans to Q No 3: ps

Ans to Q No 4: True

Ans to Q No 5: kill -9

Ans to Q No 6: time

Ans to Q No 7: parent process ID

10.9 MODEL QUESTIONS

Q 1: Explain the different attributes of a process?

Q 2: Explain any one method of system backup and recovery in Linux.

Q 3: What is process? What are the different types of process? Explain them.

Q 4: Explain the crontab file format and use of it.

Q 5: Explain the different methods of starting a process.

Q 6: Explain the working of top command.

*** ***** ***