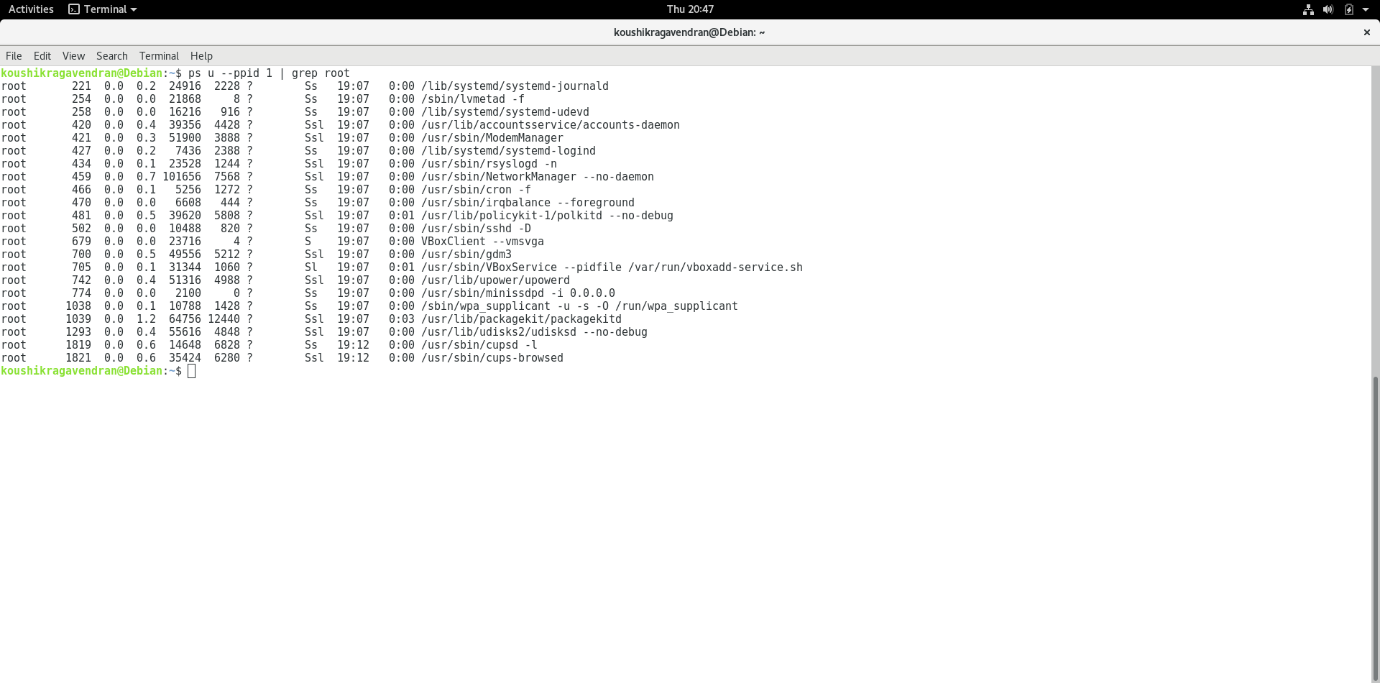
1)b)



1)c)

* **systemd-journald** is a system service that collects and stores logging data. It creates and maintains structured, indexed journals based on logging information that is received from a variety of sources. For example Kernel log messages, are stored using system-journald via kmsg.
* **lvmedtad** is a metadata caching daemon for Logical Volume manager (LVM). The daemon receives notifications from udev rules. Through these notifications lvmedtad has an up-to-date and consistent image of the volume groups available in the system.
* System-udevd: Dynamic device management. udev supplies the system software with device events, manages permissions of device nodes and may create additional symlinks in the /dev directory, or renames network interfaces.
* **Accounts-daemon** is a part of Account service, it provides interfaces for querying and manipulating user account information and an implementation of these interfaces, based on the useradd, usermod and userdel commands.
* **ModemManager** is a daemon which controls mobile broadband (2G/3G/4G) devices and connections. Whether built-in devices, USB dongles, bluetooth-paired telephones, or professional RS232/USB devices with external power supplies, ModemManager is able to prepare and configure the modems and setup connections with them.
* **systemd-logind** is a system service that manages user logins. It is responsible for keeping track of users and sessions, their processes and their idle state, Device access management for users etc
* **Rsyslog** is a way(Protocol) of producing and sending Log and Event information from Unix/Linux systems (which produces Event Logs) and Devices (Routers, Firewalls, Switches, Servers, etc) over **UDP Port 514** to a centralized Log/Event Message collector which is known as a Syslog Server.
* The **NetworkManager** daemon manages the primary network connection and other network interfaces, like Ethernet, WiFi, and Mobile Broadband devices. NetworkManager will automatically connect any network device when a connection for that device becomes available, unless that behaviour is disabled.
* **cron** is a Linux utility which schedules a command or script on your server to run automatically at a specified time and date.
* irqbalance is a Linux daemon that distributes interrupts over among the processors and cores in your computer system. The irqbalance configuration file **/etc/sysconfig/irqbalance** allows the selection of which CPU’s which may be assigned which interrupts.
* **Polkit** is a component for controlling system-wide privileges in Unix-like operating systems. It provides an organized way for non-privileged processes to communicate with privileged ones.
* **Sshd**: It is a SSH daemon providing secure [encrypted communications](https://www.lifewire.com/best-secure-email-services-4136763) between two untrusted hosts over an insecure network. It is normally started at boot-up. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution and data exchange.

3)a)

#include <stdio.h> // printf()

#include <unistd.h> // sleep(), getpid(), getppid()

#include <signal.h> // SIGCHLD flag

#include <linux/sched.h> // CLONE flags

#include <stdio.h>

void printids(char \*str) {

printf("TGID: %d of %s \n", getpid(),str); // Print the TGID of the current process

printf("PPID: %d of %s \n", getppid(),str); // Print the PPID of the current process

printf("\n\n");

sleep(1);

}

int child(char \*str)

{

printids(str);

}

int main(void)

{

int pid = getpid();

void \*pchild\_stack = malloc(1024 \* 1024);

// TODO Implement:

char \*str = "Clone";

// 1 fork() and 1 clone() to create a process //

fork();

clone(child, pchild\_stack + (1024\*1024),SIGCHLD,str);

// 1 clone() that creates a new thread //

clone(child, pchild\_stack + (1024\*1024),CLONE\_VM | CLONE\_FS | CLONE\_FILES | CLONE\_SYSVSEM | CLONE\_SIGHAND | CLONE\_THREAD | CLONE\_SETTLS | CLONE\_PARENT\_SETTID

| CLONE\_CHILD\_CLEARTID | 0,str);

char \*s;

if(getpid()==pid)

s = "Main";

else

s = "Fork";

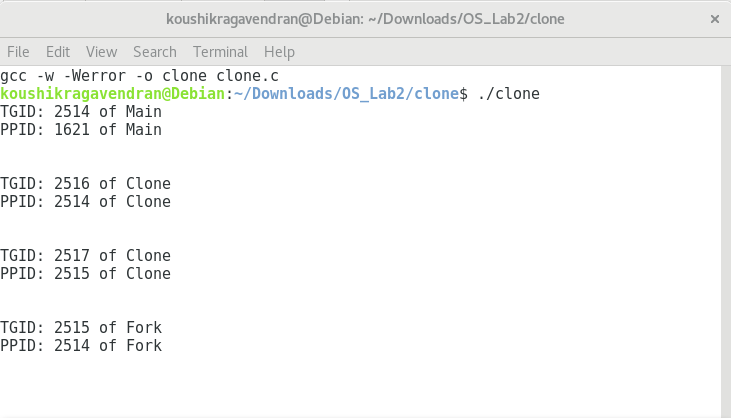
printids(s); // IDs of the main process

sleep(1000);

return 0;

}

3)b)



3)c)

The above printed ids are TGID and PPID of the processes. One clone process created from main and other clone process from the fork. Two clone thread are created by main and fork in program sharing resources from these above printed processes.



**4 c)**

The kernel module itself runs in kernel space. Loadable kernel module is an object file that contains code to extend the running kernel, or so-called base kernel, of an operating system.