ECEN5623, Real-Time Embedded Systems

Exercise #2- Service Scheduling Feasibility

UNDER THE GUIDANCE OF:

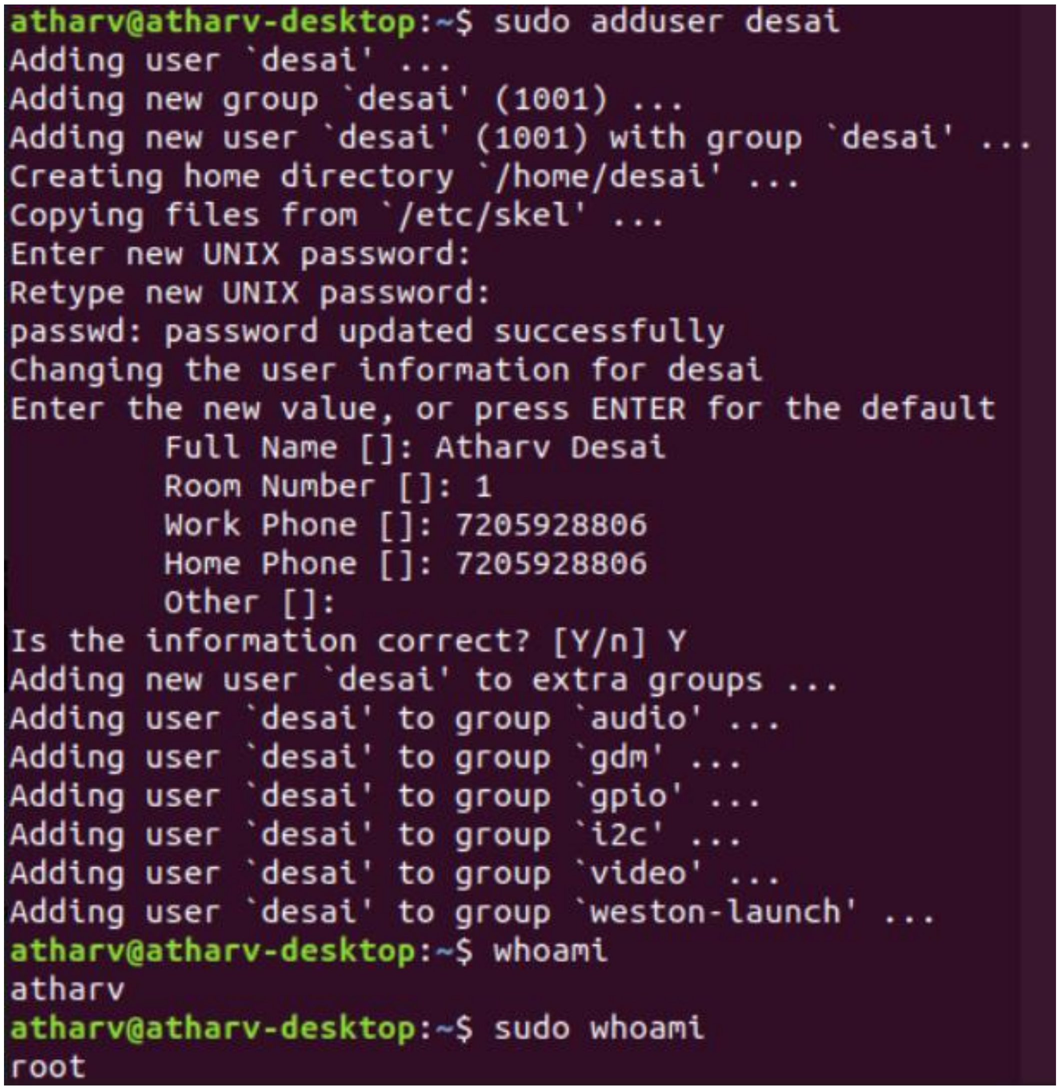
PROFESSOR TIMOTHY SCHERR

RTES Exercise 2

Question 1 :

Solution:

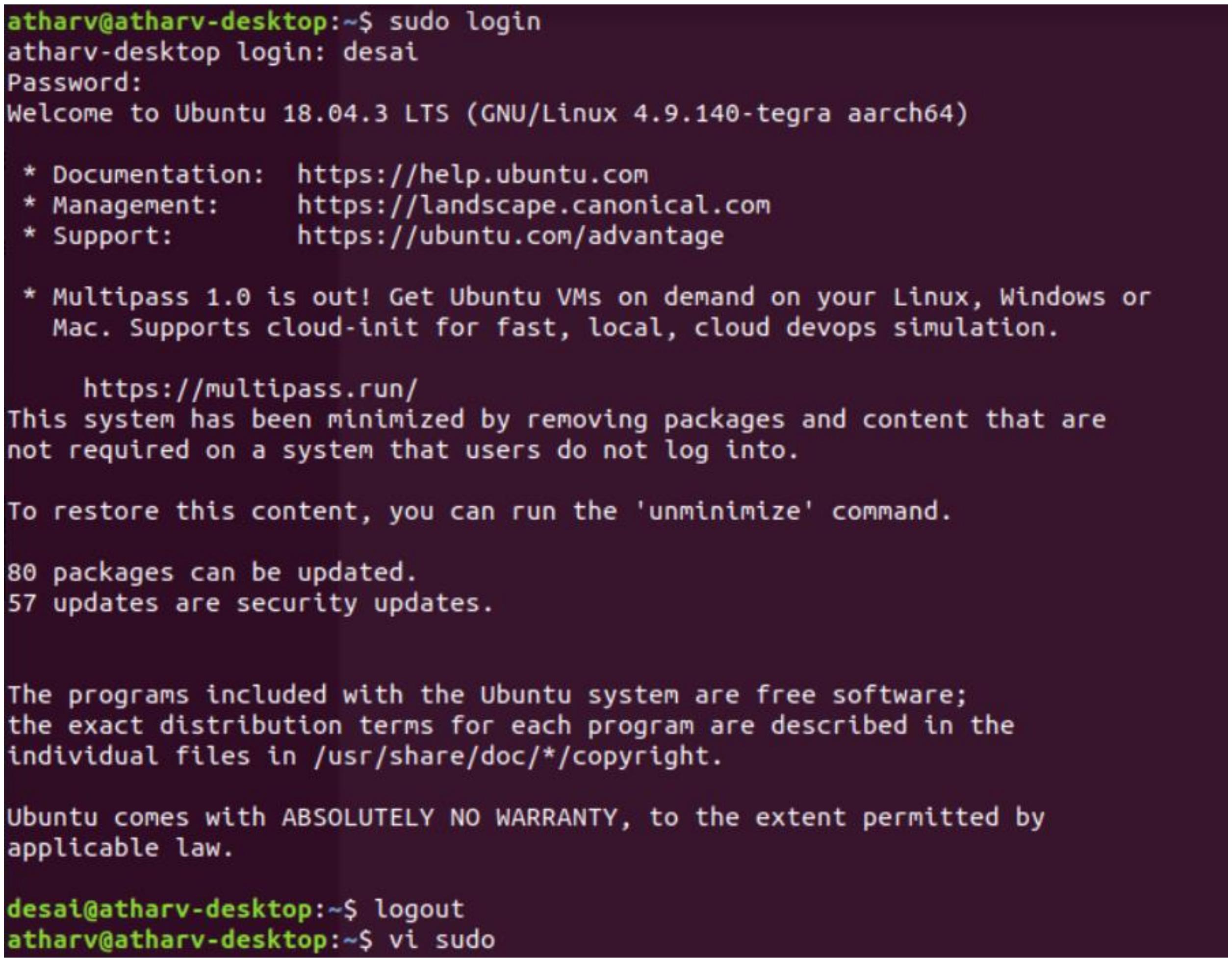
Step 1: Add User and password



```
atharv@atharv-desktop:~$ sudo adduser desai
Adding user `desai' ...
Adding new group `desai' (1001) ...
Adding new user `desai' (1001) with group `desai' ...
Creating home directory `/home/desai' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for desai
Enter the new value, or press ENTER for the default
        Full Name []: Atharv Desai
        Room Number []: 1
        Work Phone []: 7205928806
        Home Phone []: 7205928806
        Other []:
Is the information correct? [Y/n] Y
Adding new user `desai' to extra groups ...
Adding user `desai' to group `audio' ...
Adding user `desai' to group `gdm' ...
Adding user `desai' to group `gpio' ...
Adding user `desai' to group `i2c' ...
Adding user `desai' to group `video' ...
Adding user `desai' to group `weston-launch' ...
atharv@atharv-desktop:~$ whoami
atharv
atharv@atharv-desktop:~$ sudo whoami
root
```

Figure : Using adduser and whoami commands

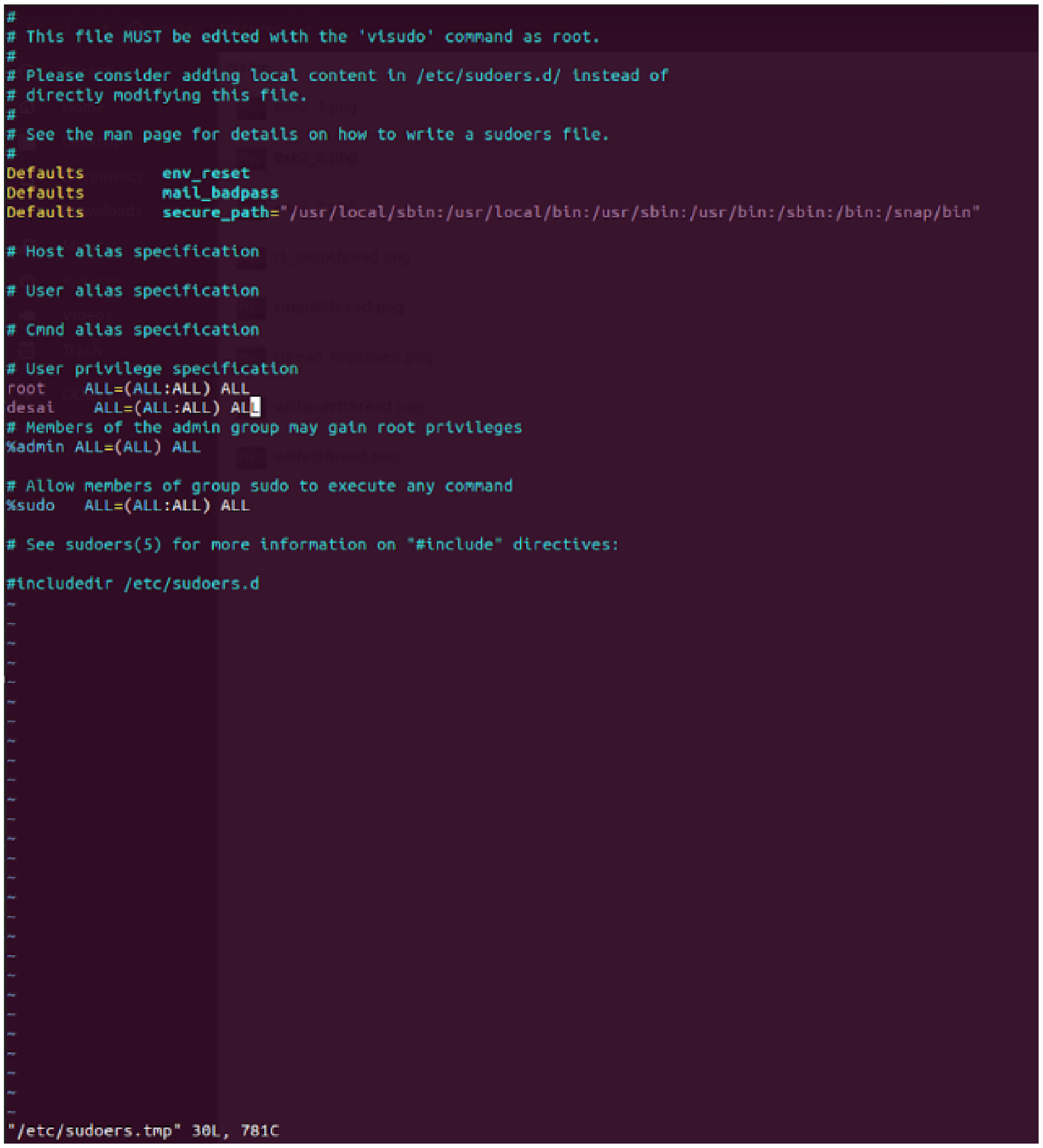
Step 2: Logging into the created new user and Logging out



```
atharv@atharv-desktop:~$ sudo login
atharv-desktop login: desai
Password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.9.140-tegra aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Multipass 1.0 is out! Get Ubuntu VMs on demand on your Linux, Windows or
   Mac. Supports cloud-init for fast, local, cloud devops simulation.

     https://multipass.run/
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

80 packages can be updated.
57 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

desai@atharv-desktop:~$ logout
atharv@atharv-desktop:~$ vi sudo
```

Figure : Using Sudo Login to log into ‘desai’ account and logging out back to ‘atharv’ root user

Step 3 : Adding username in visudo file



```
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
desai    ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"/etc/sudoers.tmp" 30L, 781C
```

Figure: Adding username “desai” in the visudo file along with root

Question 2 :

Solution:

a) Technical Paper Summary and Speculation

- ➔ This technical paper primarily discusses about the architecture and working principle of Space Shuttle Primary Avionics Software System (PASS) and allows us to delve into real-time concepts associated with it.
- ➔ Its intricate system architecture supports a congregation of objectives to assist disparate shuttle operations such as computer redundancy modules to handle the system during errors and faults.
- ➔ Because of the massive software configuration size, the system needs mass memory along with (AP-101 with 106K 32-bit words) it is divided into 8 operational sequences representing 8 operational flight stages.
- ➔ In addition, Flight Computer Operating System (FCOS) handles resource allocation as a response to the OPS sections' requests and manages I/O operations and redundancy.

b) PASS Architectural Drivers

- ➔ PASS is the cardinal module to a majority of space shuttle orbiter system functions.
- ➔ The factors which influence the architecture of the PASS are:
  - Data processing System(DBS) / IBM AP-101 General Purpose Computer(GPC)
  - Multicomputer redundancy management and synchronization.
  - Operation sequencing mode/control and man/ machine interface requirements.
  - Multiple sources of requirements definition, verification and validation and support of the PASS.

c) Operational Structure

- ➔ The operational structure main memory has a size of AP-101 (106K 32-bit words). Which is insufficient for the software (500K) containing the PASS requirements.
- ➔ The software can be said to be reliable when it handles a critical mission phase (such as ascent) be keeping the redundant GPC main memory for the complete phase to ensure that working of the system in case of main system failure.
- ➔ The mass memory feeds the software required for each OPS into the GPC main memory from the mass memory by during initiating the operational sequence. While the non-critical situations like loading of the display during on-orbit coast periods are prevented from loading to the GPC's main memory when an OPS is executing.

- ➔ OPS memory load has three parts:
  - i. Resident/System Software – OPS loads' common code/data.
  - ii. Major function base – code & data common to major app functions
  - iii. OPS overlay – applications code and unique OPS data
- ➔ The current OPS contents decide which of the three parts must be loaded to support new OPS.

d) <u>Man/ Machine Interface</u> :

- ➔ The PASS system consists of man/machine interface where the team controls and monitors orbitar avionics system by interacting with the computers.
- ➔ The multimodal OPS makes use of the OPS subordinate structures.
- ➔ OPS has major modes sub-structured into blocks linked to CRT displays.
- ➔ Switching from one mode/block to another one takes place in two ways
  - i. Manually done by keyboard input
  - ii. Software detects occurance of specific event or condition without any human intervention.
- ➔ SPEC which is the OPS sub-structure's second element is initiated when the crew member provides keyboard input.
- ➔ Once initiated, SPEC executes without any dependencies and in parallel manner along with other processes
- ➔ The Display Function (DISP) which is the third element of OPS doesn't require initialization.
- ➔ Finally, the control segments develop the OPS structure, major mode, SPEC or DISP using standardized logic blocks series.

e) <u>System Software</u> :

- ➔ The flight system software consists of small size of 15-30k & limited set of functions.
- ➔ A synchronous design strategy is adopted by the software architectures of these flight systems where a specific errand is dispatched according to a stringent timeline relative to start point of overall system cycle.
- ➔ The associated I/O needs to be strictly synchronized to the start and end of different application processes to prevent overruns at system and process levels
- ➔ The reason why a synchronous approach was adopted for the architecture of PASS system was because of a disparate number of application functions needed to be performed in the shuttle program
- ➔ Sequence and control operations accomplishment → Major components interfaces
- ➔ Management of GPC internal resources, external interfaces → FCOS

f) Flight Computer Operating System :

- ➔ GPC's internal resources were controlled and managed by FCOS along with external interfaces
- ➔ FCOS handles three main responsibilities:
   - i. Process Management: System requests and as a response, the allocation of resources is controlled by process management. It validates that the CPU resources are allocated to the task which is ready and has highest priority. A standard set of service interfaces (SVC) are made with reference to specifications that define time, or event ,priorities and frequency to handle a request.
   - ii. I/O Management : Controls the management of I/O processor resources (IOP). Each GPC containing IOP has MSC (Master Sequence Controller) and 24 Bus control Elements (BCEs) where asynochronus communication takes place between a processor and its next higher/lower priority processor at the time of I/O operations.
   - iii. DPS Configuration Management : Hardware initialization, status checks and other IOP states as well as GPC controlling is done here. Performs the transfer of program code to main memory of GPC from mass memory. It implements the overlay of program as well as modifies the contents of mass memory.

g) System Control : System control impersonates a wide range of tasks such as Configure the associated avionic data network & control DPS. SPECS also performs functions like:

- i. Resetting the MTU
- ii. Initiating memory dump
- iii. Changing and examining specific core location in memory.
- iv. Changing figuration of DPS

h) User Interface : It accelerates external control of systems or applications. Its three major functions are enunciated below:

- i. Command input processing
- ii. Input processing
- iii. Operations control and output message processing

The user interfaces supported are:

- i. Keyboard and CRT

ii. Launch Data Bus (LDB) to communicate with launch processing system

iii. Network signal Processing (NSP) to process the data and commands received from Mission Control Center at Johnson Space Station.

i) Application Software : The three applications of application software are guidance, navigation and control (GN & C). GN&C software computes the position , velocity and altitude as well as manages sensor redundancy and controls the avionics subsystems and issues the engine and affector commands for mission takeoff to landing.

1. Guidance, navigation & control :

➔ In this cyclic closed loop application, stringent timing and phasing relationship is followed by GN & C.

➔ There are 1-10 major modes included in six OPS and there are 10 SPECs and some are present in one or more of OPS

➔ The initiation, termination and sequencing of each OPS, major mode and SPEC are handled by the control segment.

➔ The principal functions are executed at 25Hz for basic vehicle system and display update at 0.25Hz

2. System Management :

➔ The system performance and the configuration of orbiter are monitored by System Management.

➔ The alert signals are generated and the crew is informed about them when the abrupt and anomalous conditions are detected while monitoring payload subsystems.

3. Vehicle Checkout :

➔ It handles the initialization and checkout of avionics system under the control of ground or flight crews

➔ It contains a special feature known as TCS (Test Control Supervisor). A maximum of three TCS commands can be executed at the same time.

➔ The configuration of associated I/O interfaces and primary functions is done into three ground checkout OPS and one in-flight checkout OPS.

The advantages of Frequency Executive are:

1) The frequency executive follows a cyclic architecture to ensure the synchronization and task execution in a defined time frame. This ensures that the execution of the tasks doesn’t get halted on one particular task which might be facing runtime error.
2) Due to a fixed order of execution following a stringent time frame, jitter dets dwindled considerably

3) The application processes can maintain synchronization by implementing redundancy management system.
4) The schedule used is predictable like fixed priority scheduling algorithm unlike using preemption. Therefore, lower priority tasks doesn't get masked by higher priority ones.

The disadvantages of Frequency Executive are:

1) Race conditions can occur during the initialization of new process since it might cause overruns for the processes at process and system levels. This happens due to asynchronization between I/O operations and intercommunications between the GPC.
2) Frequency scheduling follows non-preemptive mechanism i.e a low priority task when executing can block a much higher priority task.
3) In frequency executive method, all the possibly currently executing tasks needs analysis at the same time. Each thread is provided with dynamic or fixed scheduling priority for execution in real time scheduling.
4) There are preordained timing blocks in the cycle loop depending on mission requirements known as "Orbit Insertion Burn Stage" which has a window of 5 minutes. However, for many processes it will be difficult to be divided into blocks with 5 minute window. Therefore, creating an ideal frequency executive is difficult.

Question 3:

Solution :

A) Overview of Cyclic Executive :

➔ The cyclic executive is usually represented by a while(1) loop in main, which cycles through separate tasks that need to run.
➔ The structure may consist of several “minor cycles” inside of a single “major cycle,” where the major cycle is run over and over again.
➔ Each “task” that needs to run is allocated some time in one or more minor cycles, and it will not run again until its scheduled time comes up again.
➔ This means that there is no preemption, and all tasks are given the same priority.
➔ If you want a task to have higher priority, you could try to make it more responsive by scheduling that particular task more frequently
➔ However, but still it does not preempt another task or run until it reaches its execution time in the loop.

B) Comparison to Linux and RTOS approaches :

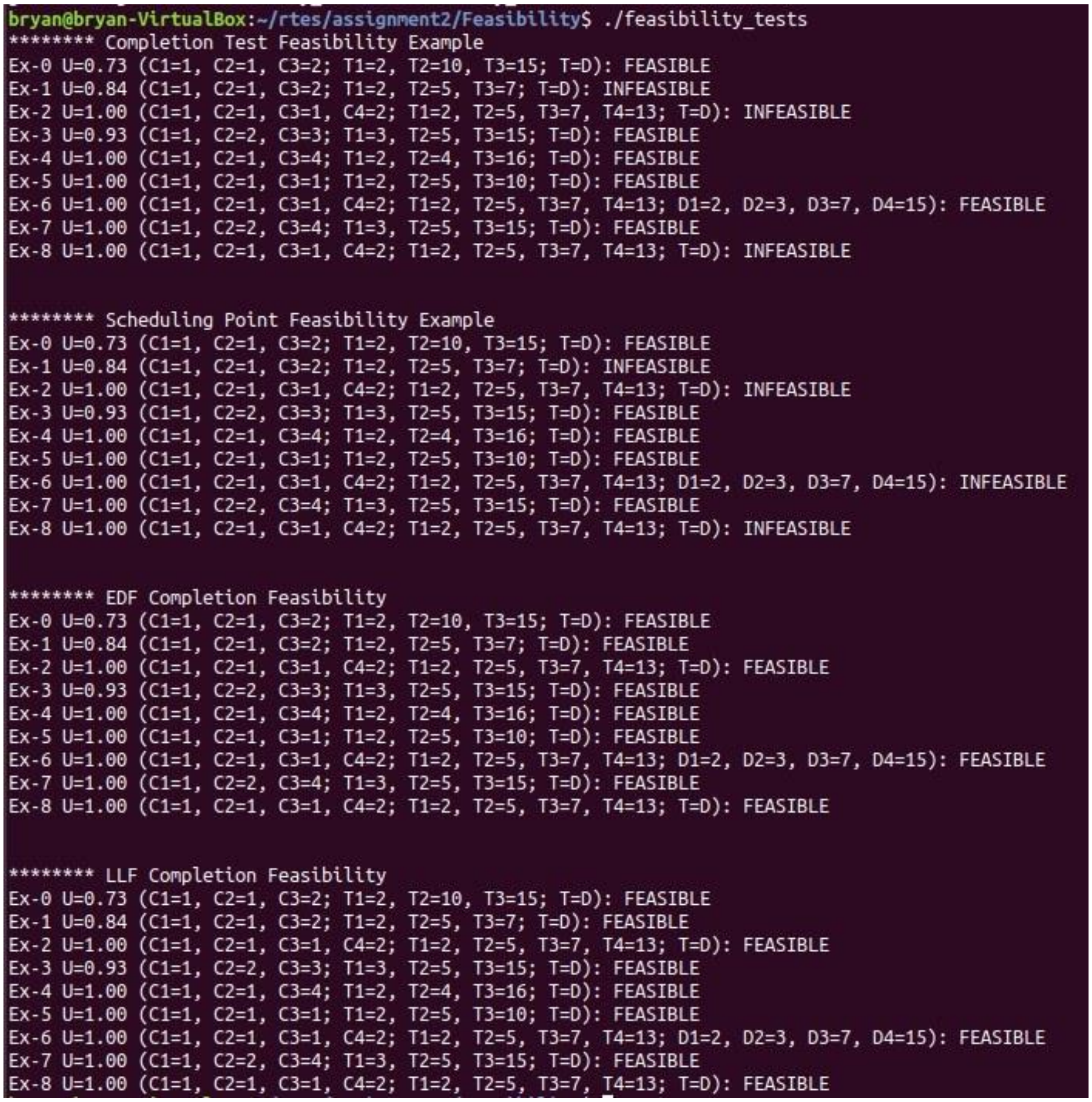
➔ There is no multitasking or multithreading, so this type of system is less susceptible to race conditions than an RTOS.
➔ There is also no overhead (memory or timing) with context switches, which means a cyclic executive can have lower memory requirements and more useful CPU utilization.
➔ An RTOS has the advantage of priority and preemption. Certain tasks can be allowed to run as soon as they are ready, preempting lower priority tasks if necessary.
➔ This provides faster and more deterministic response time to critical events/high priority tasks, at the expense of response time to lower priority tasks.
➔ RTOSes also have an advantage when trying to add an additional task to an existing system.
➔ With an RTOS, you can just add the task, give it an appropriate priority, and be confident that higher priority tasks will not be affected at all.
➔ With a cyclic executive, you would have to rethink and probably rework the whole cycle to make sure response times and periods for all your tasks are still acceptable.
➔ In this way, it is easier to create a robust system with an RTOS, and even in the event of overload, and RTOS will degrade gracefully.
➔ Linux POSIX RT threading is similar to an RTOS in that each thread has a priority, and preemption occurs, meaning higher priority tasks are run as soon as they become ready, without having to wait for lower priority tasks to finish.

- This system shares the advantage of an RTOS when adding tasks, in that adding a lower priority task will not affect the higher priority tasks at all, and in the event of a system overload, the system will degrade gracefully.
- There are two major downsides to using a Linux POSIX RT compared to a cyclic executive.
- First, Linux requires much more CPU memory and resources (more than an RTOS too), making it less ideal for low-cost or low-resource designs.
- The second major disadvantage is that the Linux OS is not actually real time, despite the “RT” threads it offers.
- This means that scheduling can be less deterministic because the OS may need to use the CPU at any time for some unspecified amount of time.

Question 4 :

Solution:

The following screenshot shows the output of the feasibility code. It contains output from the two original functions, as well as the output of the EDF and LLF functions. Each of these functions was run on all 9 examples, and the output is shown together in one screenshot.



```
bryan@bryan-VirtualBox:~/rtes/assignment2/Feasibility$ ./feasibility_tests
******** Completion Test Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=1, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; D1=2, D2=3, D3=7, D4=15): FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE


******** Scheduling Point Feasibility Example
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): INFEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=1, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; D1=2, D2=3, D3=7, D4=15): INFEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): INFEASIBLE


******** EDF Completion Feasibility
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): FEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): FEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=1, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; D1=2, D2=3, D3=7, D4=15): FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): FEASIBLE


******** LLF Completion Feasibility
Ex-0 U=0.73 (C1=1, C2=1, C3=2; T1=2, T2=10, T3=15; T=D): FEASIBLE
Ex-1 U=0.84 (C1=1, C2=1, C3=2; T1=2, T2=5, T3=7; T=D): FEASIBLE
Ex-2 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): FEASIBLE
Ex-3 U=0.93 (C1=1, C2=2, C3=3; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-4 U=1.00 (C1=1, C2=1, C3=4; T1=2, T2=4, T3=16; T=D): FEASIBLE
Ex-5 U=1.00 (C1=1, C2=1, C3=1; T1=2, T2=5, T3=10; T=D): FEASIBLE
Ex-6 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; D1=2, D2=3, D3=7, D4=15): FEASIBLE
Ex-7 U=1.00 (C1=1, C2=2, C3=4; T1=3, T2=5, T3=15; T=D): FEASIBLE
Ex-8 U=1.00 (C1=1, C2=1, C3=1, C4=2; T1=2, T2=5, T3=7, T4=13; T=D): FEASIBLE
```

Figure: Output of Feasibility Code

Examples 0-3

For the first 4 examples (examples 0-3), the feasibility code and Cheddar agreed on all examples. Examples 0 and 3 were considered feasible, while examples 1 and 2 are infeasible. The output of the feasibility code can be seen in the screenshot at the beginning of the "Question 4 section." For the comparison in Cheddar, I used a RM scheduling policy, and the screenshots of each can be seen in the screenshots folder submitted along with this report.

Example 4

RM

The RM scheduling policy was successful for this example. There is 100% utilization, but that is ok, even with RM, because the tasks are harmonic. Both the code output and Cheddar reported that the tasks were schedulable with RM.

EDF

EDF was successful for this example. Both the feasibility code and Cheddar agreed that the tasks were schedulable with EDF, and actually ended up with the same schedule as the RM scheduler!

LLF

LLF was successful for this example. Both the feasibility code and Cheddar agreed that the tasks were schedulable with LLF, and actually ended up with the same schedule as the RM scheduler!

Example 5

RM

The RM scheduling policy was successful for this example. This is another example of RM with 100% utilization, so at first glance it may not seem schedulable, but it turns out it is! Even though it fails the RM LUB test, it is schedulable with 100% utilization. Both the code output and Cheddar reported that the tasks were schedulable with RM.

EDF

EDF is also successful for this example. Both the feasibility code and Cheddar agreed that the tasks were schedulable with EDF, and actually ended up with the same schedule as the RM scheduler!

LLF

LLF is also successful for this example. Both the feasibility code and Cheddar agreed that the tasks were schedulable with LLF, and actually ended up with the same schedule as the RM scheduler!

Example 6

RM

RM cannot be used on this example because the deadline and period are not the same. This violates one of the assumptions in the RM theory that the, so we cannot assume that the RM algorithms apply. Cheddar confirms this, by reporting that "tasks must have period equal to deadline : can not apply selected scheduler with this task set."

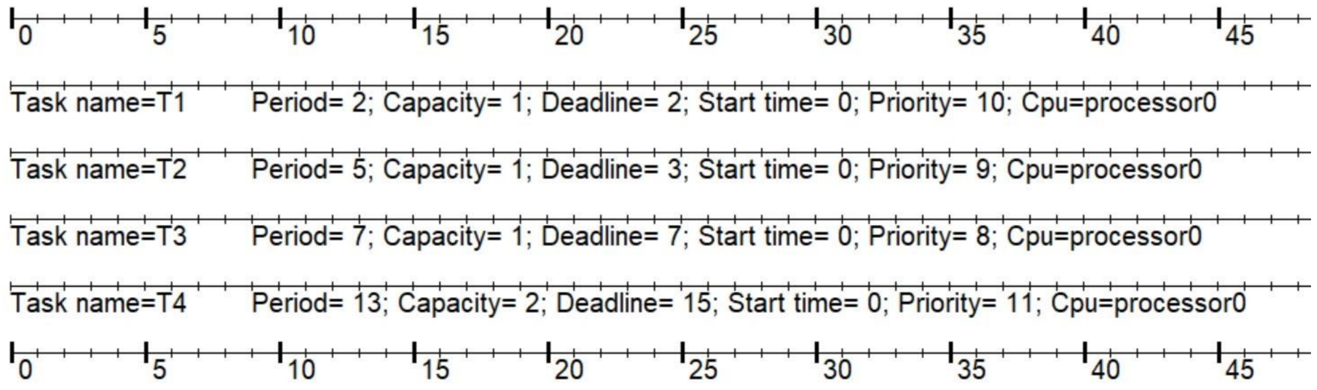
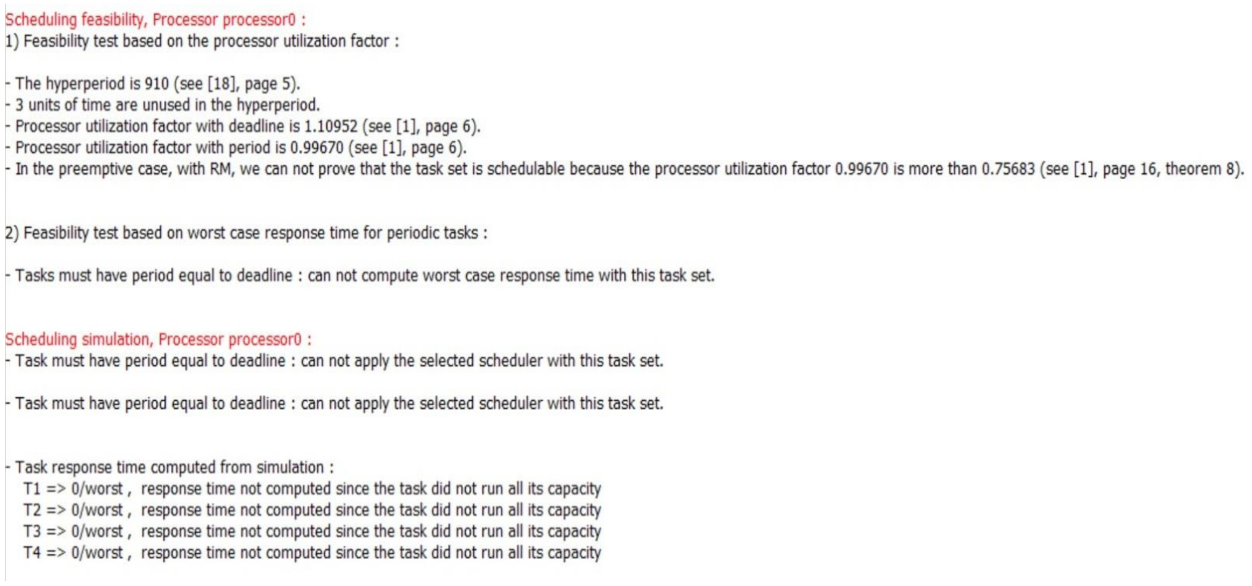

0
5
10
15
20
25
30
35
40
45
Task name=T1 Period= 2; Capacity= 1; Deadline= 2; Start time= 0; Priority= 10; Cpu=processor0
Task name=T2 Period= 5; Capacity= 1; Deadline= 3; Start time= 0; Priority= 9; Cpu=processor0
Task name=T3 Period= 7; Capacity= 1; Deadline= 7; Start time= 0; Priority= 8; Cpu=processor0
Task name=T4 Period= 13; Capacity= 2; Deadline= 15; Start time= 0; Priority= 11; Cpu=processor0

Figure: RM cannot be scheduled in Cheddar



```
Scheduling feasibility, Processor processor0 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 3 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.10952 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (see [1], page 6).
- In the preemptive case, with RM, we can not prove that the task set is schedulable because the processor utilization factor 0.99670 is more than 0.75683 (see [1], page 16, theorem 8).


2) Feasibility test based on worst case response time for periodic tasks :

- Tasks must have period equal to deadline : can not compute worst case response time with this task set.


Scheduling simulation, Processor processor0 :
- Task must have period equal to deadline : can not apply the selected scheduler with this task set.

- Task must have period equal to deadline : can not apply the selected scheduler with this task set.


- Task response time computed from simulation :
   T1 => 0/worst ,  response time not computed since the task did not run all its capacity
   T2 => 0/worst ,  response time not computed since the task did not run all its capacity
   T3 => 0/worst ,  response time not computed since the task did not run all its capacity
   T4 => 0/worst ,  response time not computed since the task did not run all its capacity
```

Figure: RM does not apply because task deadlines are not equal to task periods

EDF

We couldn't even try with RM, but EDF works on this example. CPU utilization is close to fully loaded (99.67%), but both the feasibility code and Cheddar agreed that the tasks were schedulable with EDF.

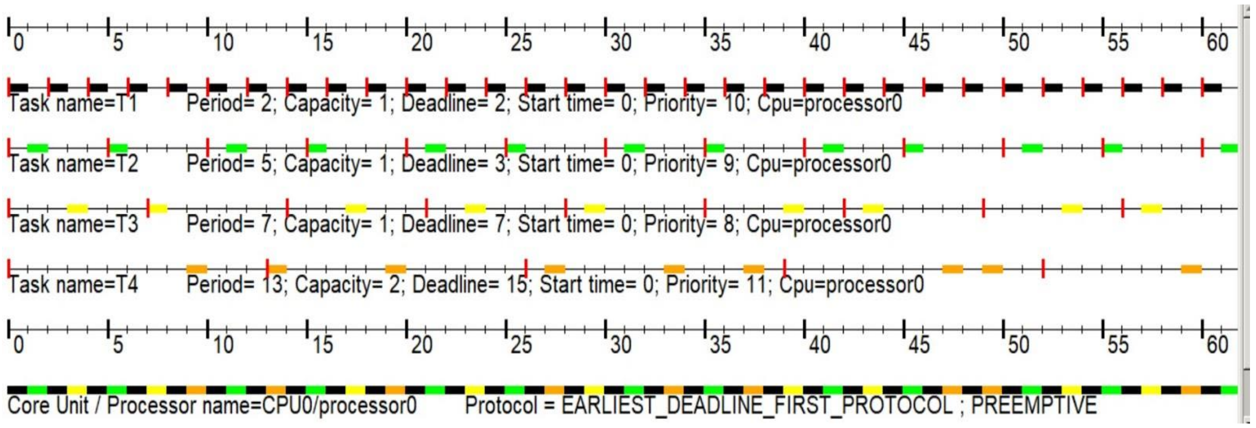
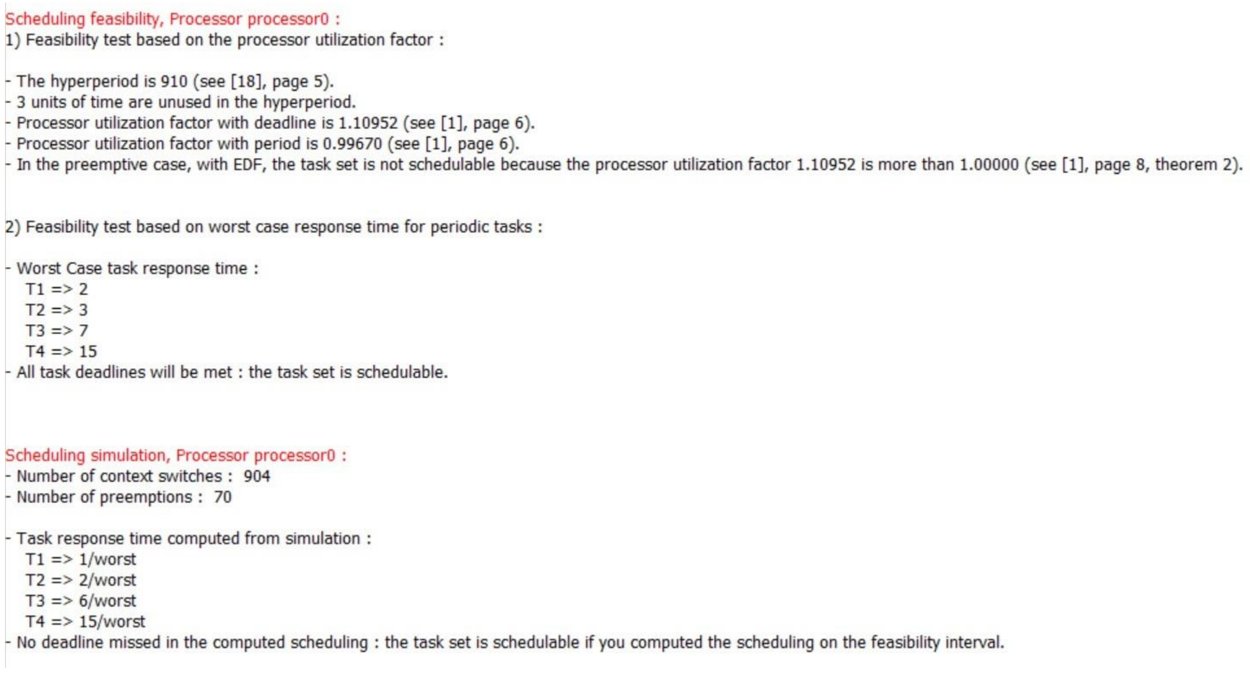


Figure: Example 6 EDF schedule



```
Scheduling feasibility, Processor processor0 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 3 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.10952 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (see [1], page 6).
- In the preemptive case, with EDF, the task set is not schedulable because the processor utilization factor 1.10952 is more than 1.00000 (see [1], page 8, theorem 2).


2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :
   T1 => 2
   T2 => 3
   T3 => 7
   T4 => 15
- All task deadlines will be met : the task set is schedulable.


Scheduling simulation, Processor processor0 :
- Number of context switches :  904
- Number of preemptions :  70

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 2/worst
   T3 => 6/worst
   T4 => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Figure: Example 6 EDF Cheddar analysis

LLF

LLF also works on this example. CPU utilization again is close to fully loaded (99.67%), but both the feasibility code and Cheddar agreed that the tasks were schedulable with LLF.

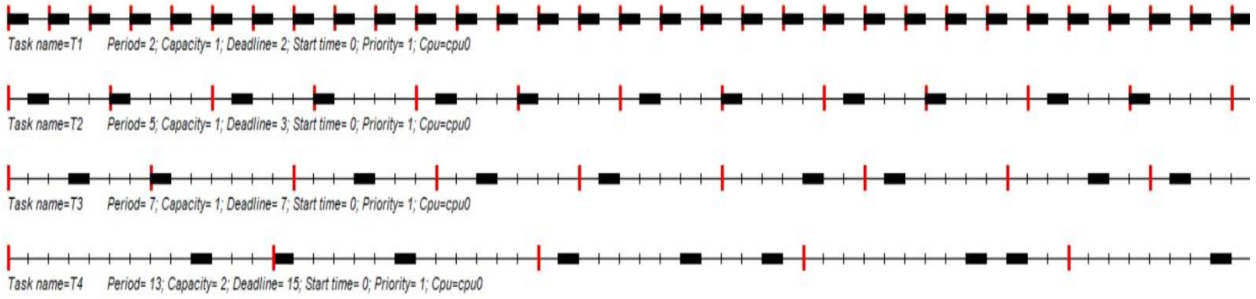
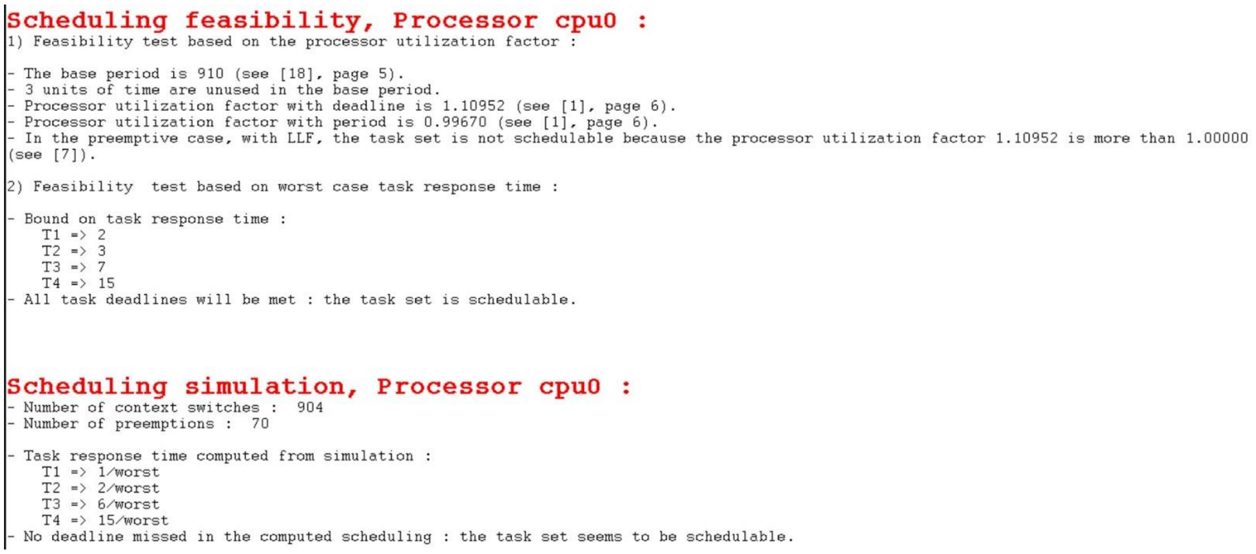


Figure: Example 6 LLF schedule



```
Scheduling feasibility, Processor cpu0 :
1) Feasibility test based on the processor utilization factor :

- The base period is 910 (see [18], page 5).
- 3 units of time are unused in the base period.
- Processor utilization factor with deadline is 1.10952 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (see [1], page 6).
- In the preemptive case, with LLF, the task set is not schedulable because the processor utilization factor 1.10952 is more than 1.00000
(see [7]).

2) Feasibility  test based on worst case task response time :

- Bound on task response time :
    T1 => 2
    T2 => 3
    T3 => 7
    T4 => 15
- All task deadlines will be met : the task set is schedulable.



Scheduling simulation, Processor cpu0 :
- Number of context switches :  904
- Number of preemptions :  70

- Task response time computed from simulation :
    T1 => 1/worst
    T2 => 2/worst
    T3 => 6/worst
    T4 => 15/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.
```

Figure: Example 6 LLF Cheddar analysis

Example 7

RM

The RM scheduling policy was successful for this example. Even though the CPU utilization was at 100%, both the feasibility code and Cheddar agreed that the tasks were schedulable with RM.

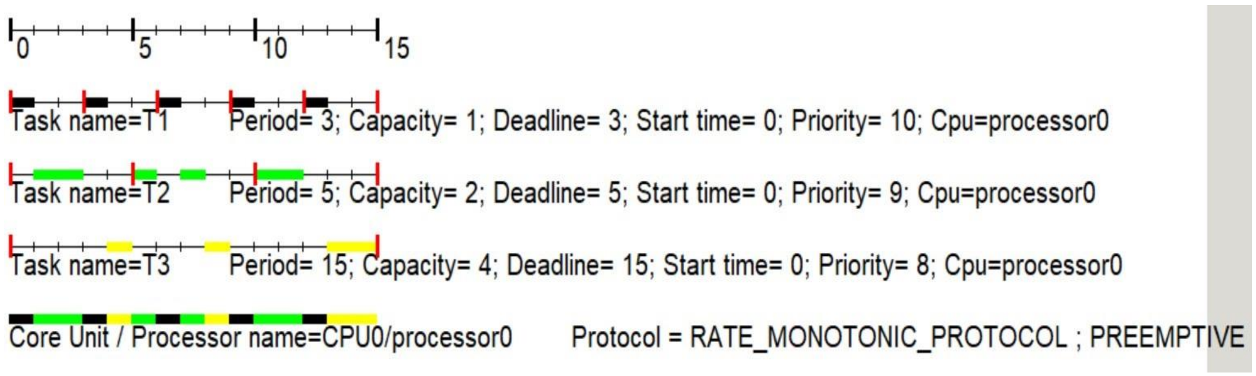
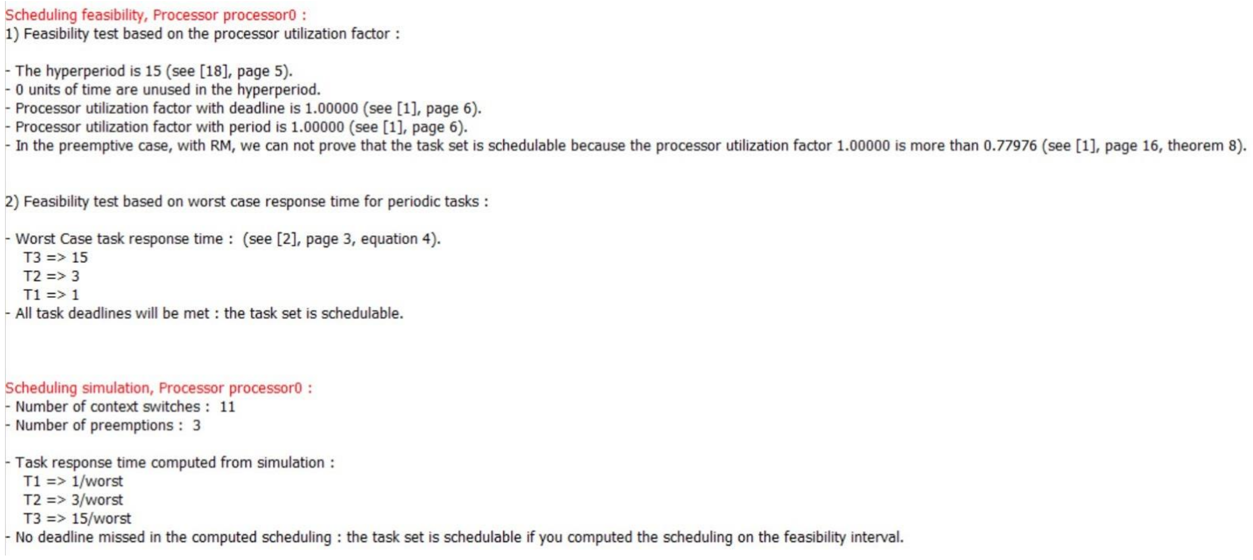


Figure: Example 7 RM schedule



```
Scheduling feasibility, Processor processor0 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 15 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.00000 (see [1], page 6).
- Processor utilization factor with period is 1.00000 (see [1], page 6).
- In the preemptive case, with RM, we can not prove that the task set is schedulable because the processor utilization factor 1.00000 is more than 0.77976 (see [1], page 16, theorem 8).


2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :  (see [2], page 3, equation 4).
   T3 => 15
   T2 => 3
   T1 => 1
- All task deadlines will be met : the task set is schedulable.



Scheduling simulation, Processor processor0 :
- Number of context switches :  11
- Number of preemptions :  3

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 3/worst
   T3 => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Figure: Example 7 RM Cheddar analysis

EDF

EDF is also successful for this example. Both the feasibility code and Cheddar agreed that the tasks were schedulable with EDF, and actually ended up with the same schedule as the RM scheduler!

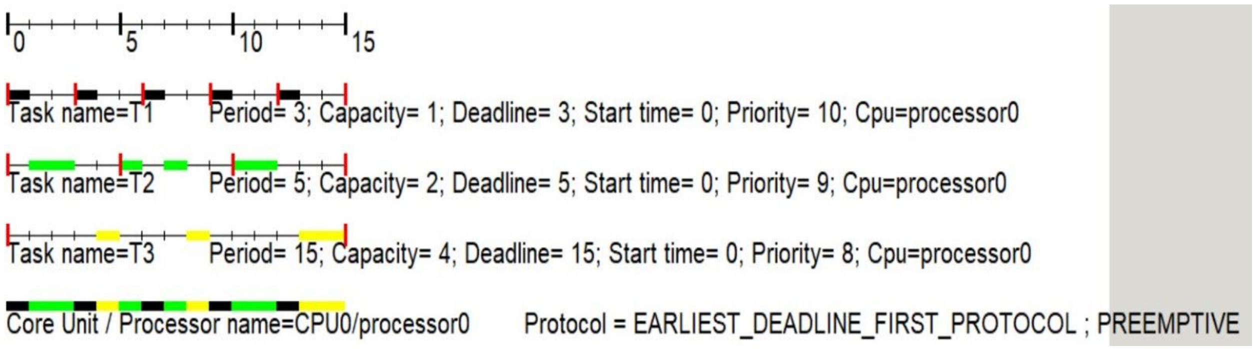
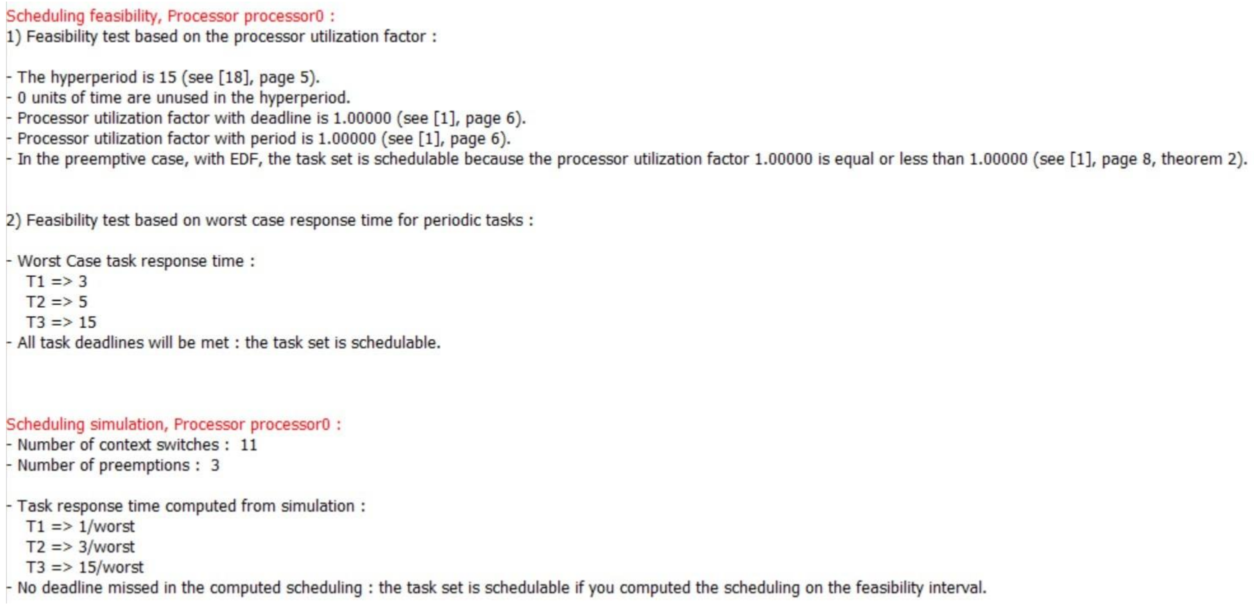


Figure: Example 7 EDF schedule



```
Scheduling feasibility, Processor processor0 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 15 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.00000 (see [1], page 6).
- Processor utilization factor with period is 1.00000 (see [1], page 6).
- In the preemptive case, with EDF, the task set is schedulable because the processor utilization factor 1.00000 is equal or less than 1.00000 (see [1], page 8, theorem 2).


2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :
   T1 => 3
   T2 => 5
   T3 => 15
- All task deadlines will be met : the task set is schedulable.



Scheduling simulation, Processor processor0 :
- Number of context switches :  11
- Number of preemptions :  3

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 3/worst
   T3 => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Figure: Example 7 EDF Cheddar analysis

LLF

LLF is also successful for this example. Both the feasibility code and Cheddar agreed that the tasks were schedulable with LLF, and actually ended up with the same schedule as the RM scheduler!

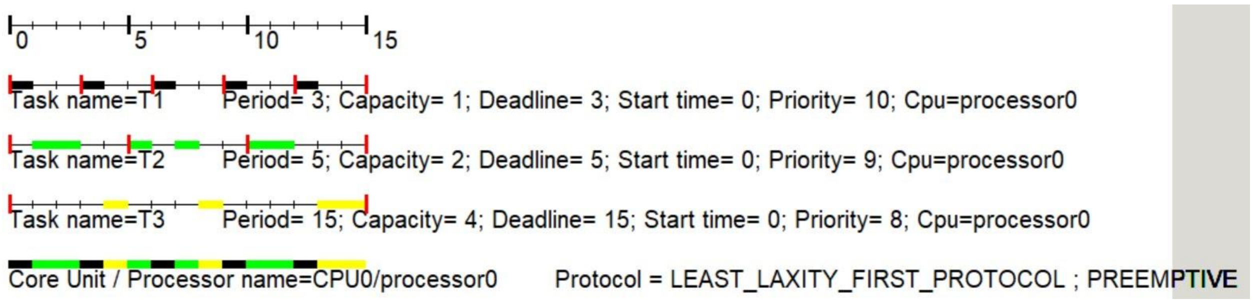
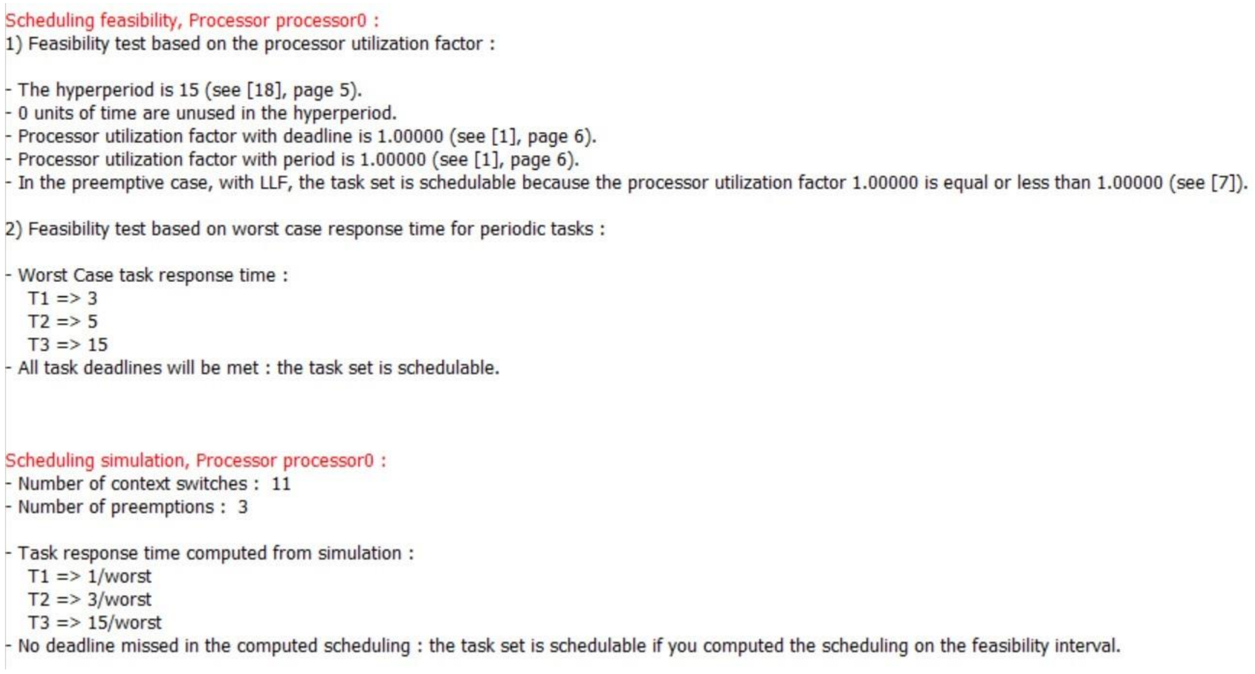


Figure: Example 7 LLF schedule



```
Scheduling feasibility, Processor processor0 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 15 (see [18], page 5).
- 0 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 1.00000 (see [1], page 6).
- Processor utilization factor with period is 1.00000 (see [1], page 6).
- In the preemptive case, with LLF, the task set is schedulable because the processor utilization factor 1.00000 is equal or less than 1.00000 (see [7]).

2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :
   T1 => 3
   T2 => 5
   T3 => 15
- All task deadlines will be met : the task set is schedulable.



Scheduling simulation, Processor processor0 :
- Number of context switches :  11
- Number of preemptions :  3

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 3/worst
   T3 => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Figure: Example 7 LLF Cheddar analysis

Example 8

RM

RM does not work for this example! This example is close to the same as example 6, with the difference that the deadlines are now equal to the periods of the tasks, which allow us to use RM. It is pretty close, but as can be seen, some deadlines are missed. This means it is not schedulable with RM, but at least there is only one task (the lowest priority one) that is missing deadlines.

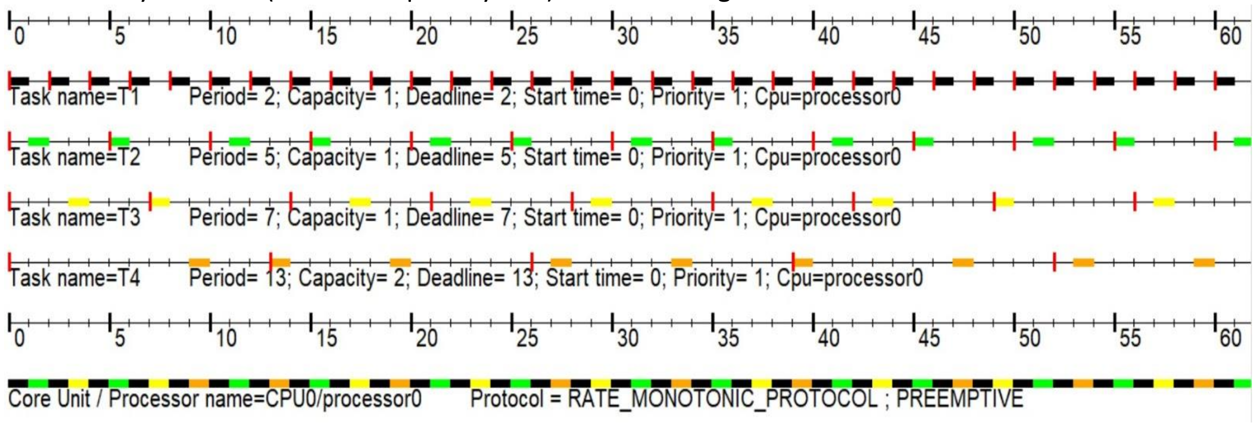
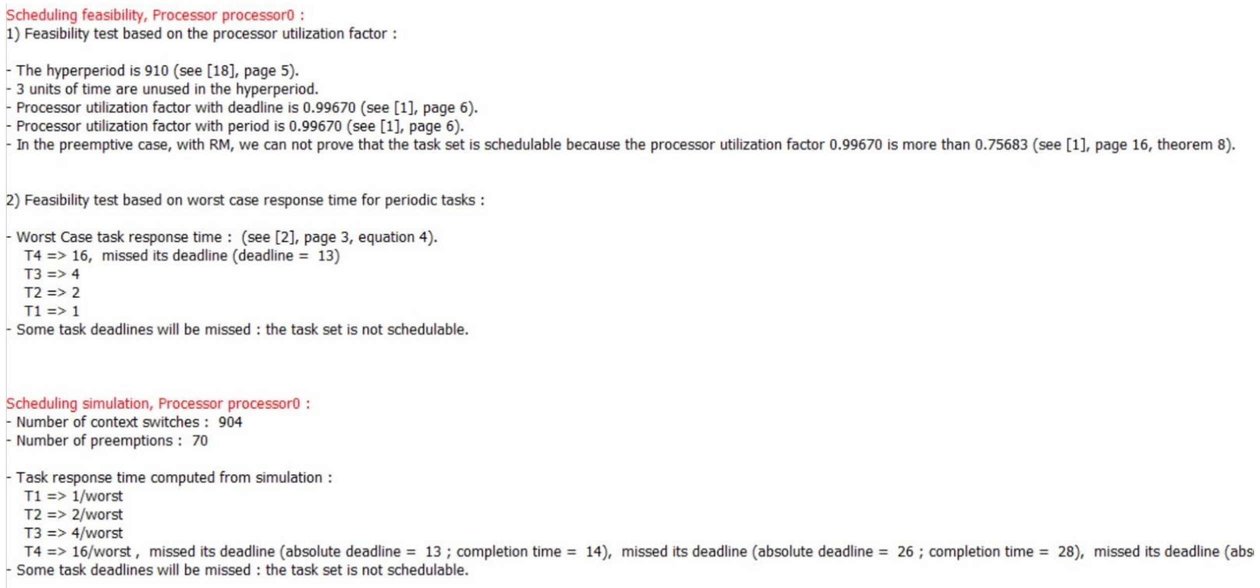


Figure: Example 8 RM schedule



```
Scheduling feasibility, Processor processor0 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 3 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.99670 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (see [1], page 6).
- In the preemptive case, with RM, we can not prove that the task set is schedulable because the processor utilization factor 0.99670 is more than 0.75683 (see [1], page 16, theorem 8).


2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :  (see [2], page 3, equation 4).
   T4 => 16,  missed its deadline (deadline =  13)
   T3 => 4
   T2 => 2
   T1 => 1
- Some task deadlines will be missed : the task set is not schedulable.


Scheduling simulation, Processor processor0 :
- Number of context switches :  904
- Number of preemptions :  70

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 2/worst
   T3 => 4/worst
   T4 => 16/worst ,  missed its deadline (absolute deadline =  13 ; completion time =  14),  missed its deadline (absolute deadline =  26 ; completion time =  28),  missed its deadline (abs
- Some task deadlines will be missed : the task set is not schedulable.
```

Figure: Example 8 RM Cheddar analysis

EDF

This example is a case where EDF works even though RM doesn't. This is because RM was pretty close, but doesn't quite utilize the processor in a way that allows everything to be scheduled. Compare the schedule in the screenshot below to the one from the RM above. Time slot 7 is key here. T3 was just made ready to run again, and under RM it of course runs. But this won't leave enough time for T4 to meet its deadline. Just a simple switch here of allowing T4 to run instead of T3 (because T4 has an earlier deadline than T3) allows T4 to meet its deadline, and there is still enough laxity in T3 for it to be scheduled a little later in its period. In this way, when the priority is shifted a bit so that T4 can get a bit more time to run when its deadline is coming up, all tasks can be scheduled successfully. Both the feasibility code and Cheddar agreed that the tasks were schedulable with EDF

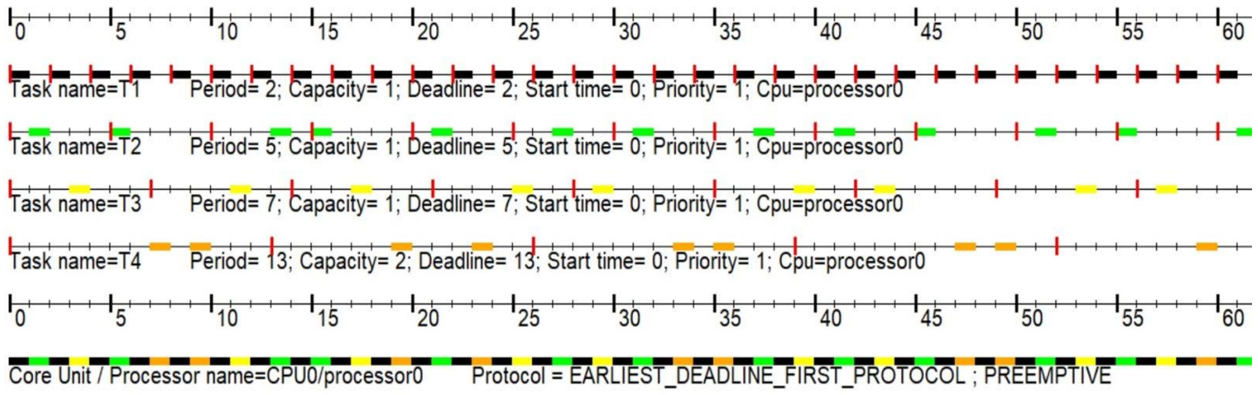


Figure: Example 8 EDF schedule

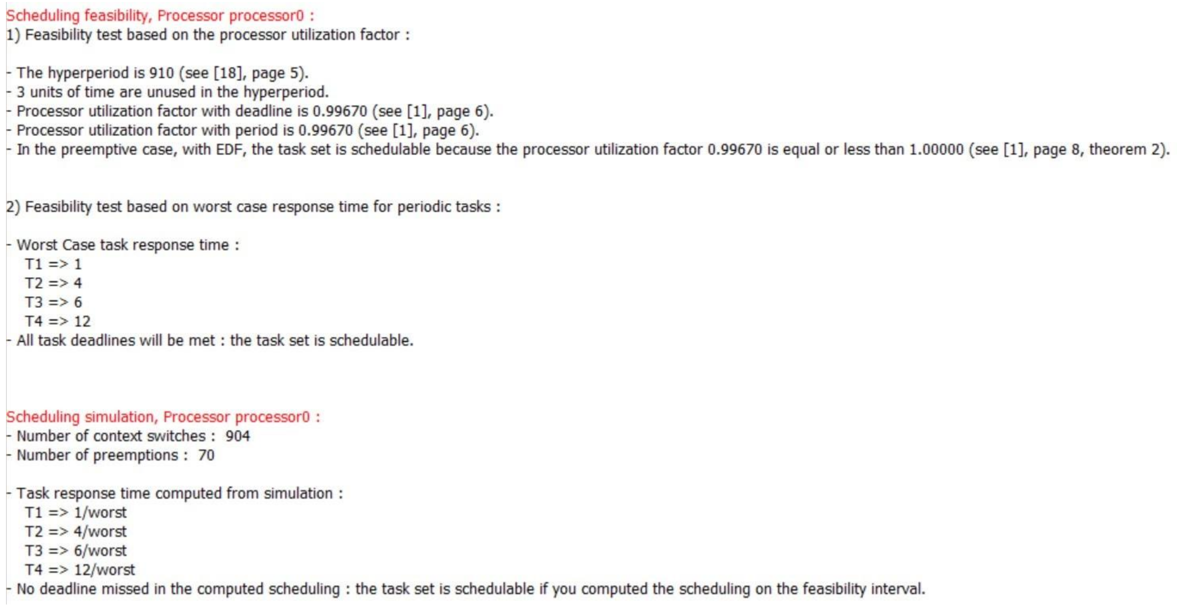

```
Scheduling feasibility, Processor processor0 :
1) Feasibility test based on the processor utilization factor :

- The hyperperiod is 910 (see [18], page 5).
- 3 units of time are unused in the hyperperiod.
- Processor utilization factor with deadline is 0.99670 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (see [1], page 6).
- In the preemptive case, with EDF, the task set is schedulable because the processor utilization factor 0.99670 is equal or less than 1.00000 (see [1], page 8, theorem 2).


2) Feasibility test based on worst case response time for periodic tasks :

- Worst Case task response time :
   T1 => 1
   T2 => 4
   T3 => 6
   T4 => 12
- All task deadlines will be met : the task set is schedulable.


Scheduling simulation, Processor processor0 :
- Number of context switches :  904
- Number of preemptions :  70

- Task response time computed from simulation :
   T1 => 1/worst
   T2 => 4/worst
   T3 => 6/worst
   T4 => 12/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Figure: Example 8 EDF Cheddar analysis

LLF

Similar to the logic for EDF, LLF also works for this example even though RM doesn't. All that is needed is a bit of reprioritization when T4 is getting closer to its deadline, and then everything fits. Both the feasibility code and Cheddar agreed that the tasks were schedulable with LLF.

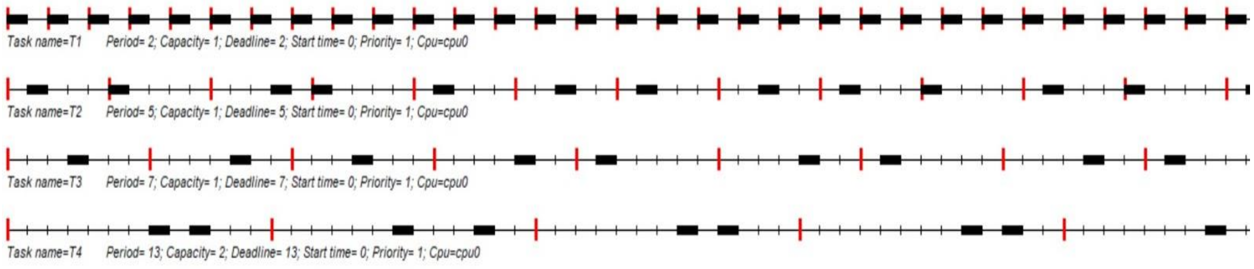
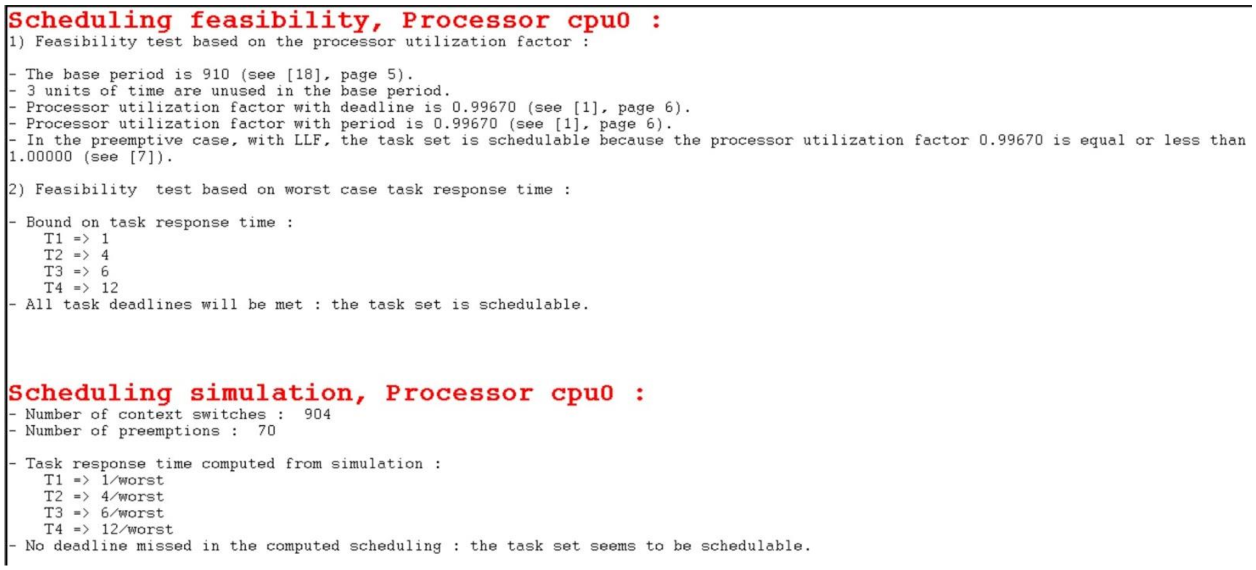


Figure: Example 8 LLF schedule



```
Scheduling feasibility, Processor cpu0 :
1) Feasibility test based on the processor utilization factor :

- The base period is 910 (see [18], page 5).
- 3 units of time are unused in the base period.
- Processor utilization factor with deadline is 0.99670 (see [1], page 6).
- Processor utilization factor with period is 0.99670 (see [1], page 6).
- In the preemptive case, with LLF, the task set is schedulable because the processor utilization factor 0.99670 is equal or less than
1.00000 (see [7]).

2) Feasibility  test based on worst case task response time :

- Bound on task response time :
    T1 => 1
    T2 => 4
    T3 => 6
    T4 => 12
- All task deadlines will be met : the task set is schedulable.



Scheduling simulation, Processor cpu0 :
- Number of context switches :  904
- Number of preemptions :  70

- Task response time computed from simulation :
    T1 => 1/worst
    T2 => 4/worst
    T3 => 6/worst
    T4 => 12/worst
- No deadline missed in the computed scheduling : the task set seems to be schedulable.
```

Figure: Example 8 LLF Cheddar analysis

Question 5 :

Solution :

The 3 assumptions documented in Liu Layland paper are:

i. The requests for all the tasks with hard deadlines are periodic in nature and there exists a constant rate of request.
ii. The tasks are mutually exclusive and independent of each other i.e. the initiation and completion of a particular task do not depend on other tasks.
iii. Run-time for a particular task is constant for that task and doesn't vary with time. Where, run-time is the total time a processor takes to perform a task without any interruption.

The constraints made on RM LUB derivation are:

i. The time period of the task is equal to the deadline of the task. Therefore, it is mandatory that the task should be provided response before its next occurrence. This will simplify the mathematical calculations. There can be placed some buffering hardware for each peripheral function to complete the task before next occurrence.
ii. Scheduling policies such as preemptive, fixed-priority and run-to-completion scheduling policy must be used. This is because rate monotonic scheduling uses pre-emptive or fixed priority tasks only. Therefore, non-preemptive and dynamic tasks are not permitted.
iii. Context switching refers to the time taken by the CPU to switch between lower priority task to higher priority task on its arrival. The derivation of RM LUB can become complicated if this switching time is taken into consideration for its derivation. Therefore, to prevent complexity, this time is overlooked.
iv. The CPU utilization factor is U = m (2^(1/m) -1). This equation is valid only when Ti/Tj<=2 i.e the ratio of any 2 of the tasks should be less than 2.

The three key derivation steps in RM LUB derivation which I found tricky were:

i. In Liu Layland paper, the Theorem 3 of Achievable Processor Utilization states that for a set of two tasks with fixed priority assignment, the least upper bound to processor utilization factor is U = 2 x (2^(1/2) -1). While explaining the case of run-time C1 short enough condition, I didn't understand how did we get the equation:

$$C_1 \leq T_2 - T_1 \lfloor T_2/T_1 \rfloor$$

However, speculating through the below two-service diagram taken from RTECS, I was able to visualize the concept. From, that, I could infer the further calculations done to evaluate the value of C2.

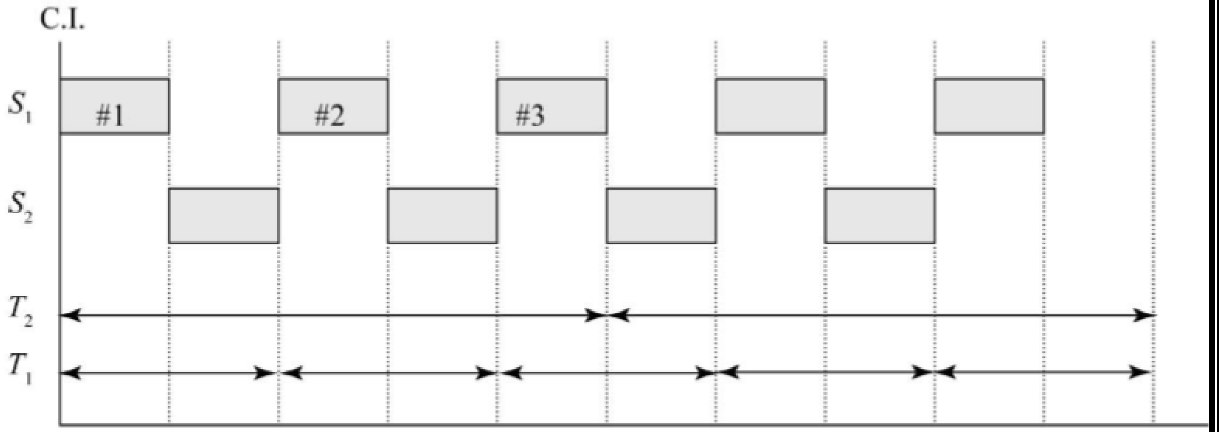


Figure : Two service example to derive RM LUB

ii. The generalized case of the utility can be obtained from the following equation

$$U = \frac{C_1}{T_1} + \frac{C_2}{T_2}$$

On substituting the value of C2 in the above equation, we get,

$$U = \frac{C_1}{T_1} + \frac{\left\lfloor T_2 - C_1 \lceil T_2 / T_1 \rceil \right\rfloor}{T_2}$$

Further simplifying the equation and evaluating it in terms of T2 and T1 we get,

$$U = 1 - (T_1 / T_2)\left[\lceil T_2 / T_1 \rceil - (T_2 / T_1)\right]\left[(T_2 / T_1) - \lfloor T_2 / T_1 \rfloor\right]$$

The fractional interference f and the Interference I is given as

$$f = (T_2 / T_1) - \lfloor T_2 / T_1 \rfloor.$$

$$I = \lfloor T_2 / T_1 \rfloor$$

The least upper bound for utility can be calculated when its value is reduced. However, its obscure to understand that to reduce the utility, when I needs to be decreased and at the same it, we weren't able to figure out the reason why I was substituted to 1.

iii. Consider the evaluated utility equation:

$$U = (T_1 / T_2)\lfloor T_2 / T_1 \rfloor + C_1\left[(1/T_1) - (1/T_2)\lfloor T_2 / T_1 \rfloor\right]$$

Here when T2 > T1, U increases monotonically with C1

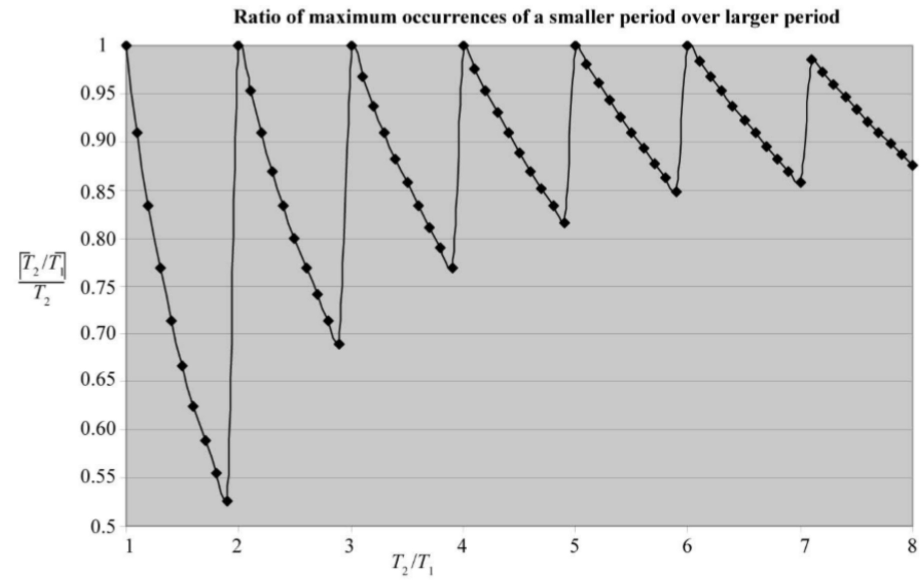


Figure : Relationship for T2 and T1 for Case 2.

The term (1/T2) [T2 / T1] will be always smaller than (1/T2) since T1 will be always smaller than T2. To simplify the calculation, there is an assumption made that T1=1 and T2 is given value which is greater than 1 till infinity. However, I couldn't decipher why T1 is considered as 1 and not some other value. If that is understood, then we can comprehend the flow of derivation and it won't be obscure.

References:-

1) http://mercury.pr.erau.edu/~siewerts/cec450/documents/Papers/liu_layland.pdf
2) Scheduling Analysis done using Cheddar software.
3) https://www.digitalocean.com/community/tutorials/how-to-create-a-sudo-user-on-ubuntuquickstart
4) Linux Man Pages
5) Real Time Embedded Components and Systems – using Linux and RTOS by Sam Seiwart
6) Building safety-critical real-time systems with reusable cyclic executives - J. Zamorano, A. Alonso, J.A. de la Puente from http://dx.doi.org/10.1016/S0967-0661(97)00088-9
7) Architecture of the Space Shuttle Primary Avionics Software System [also available on Canvas, "shuttle_paper.pdf"], by Gene Carlow
8) https://www.geeksforgeeks.org/program-to-find-lcm-of-two-numbers/