Design and Engineering of Computer Systems

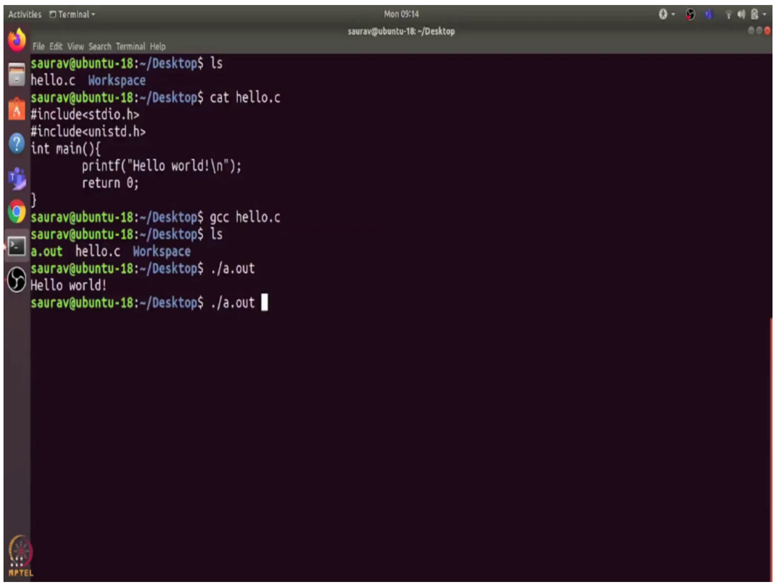
Professor Mythili Vutukuru
Computer Science and Engineering
Indian Institute of Technology, Bombay

Lecture 7

Lifecycle of C Program

Hi students, in this video, we will see what happens when we run a C program. So, here you can see that I have written a hello dot c file on the desktop. Let us open a terminal.

(Refer Slide Time: 0:28)



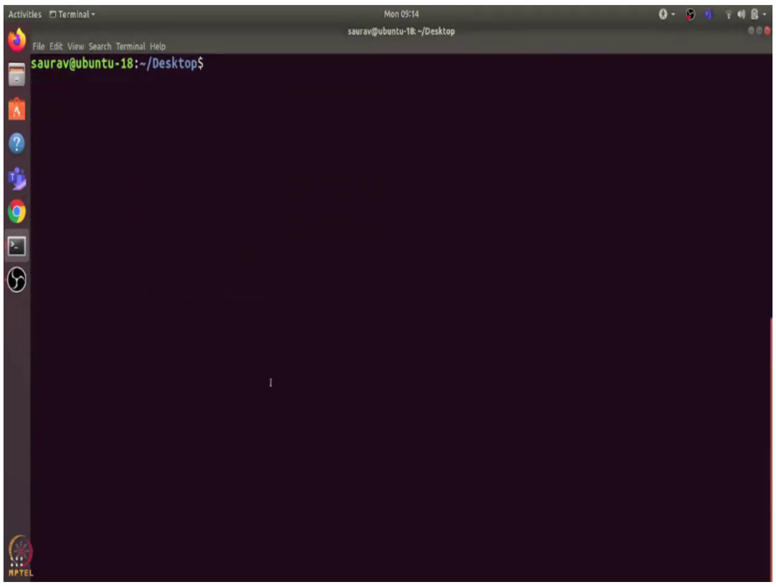
```
saurav@ubuntu-18:~/Desktop$ ls
hello.c  Workspace
saurav@ubuntu-18:~/Desktop$ cat hello.c
#include<stdio.h>
#include<unistd.h>
int main(){
        printf("Hello world!\n");
        return 0;
}
saurav@ubuntu-18:~/Desktop$ gcc hello.c
saurav@ubuntu-18:~/Desktop$ ls
a.out  hello.c  Workspace
saurav@ubuntu-18:~/Desktop$ ./a.out
Hello world!
saurav@ubuntu-18:~/Desktop$ ./a.out
```

So, this is the hello dot c file and let us have a look at its contents. So, it is a very simple C program, it just prints hello world and then it returns. So, the first thing that we need to do is, we need to compile this C program and what is compilation? So, this program is something which we can understand, but CPU cannot understand it as is. And we need to convert this program into instructions that a CPU can understand. So, this is exactly what a compiler program like GCC does.

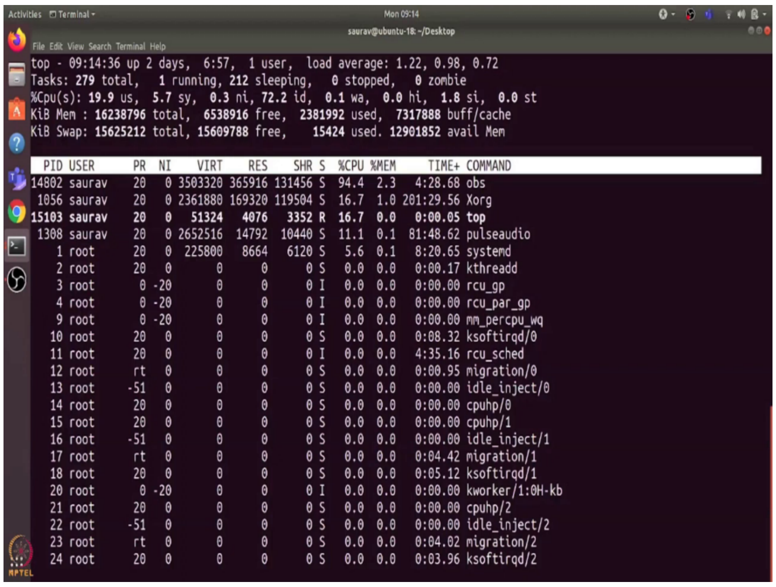
It creates an executable file called a dot out, which contains instructions based on the underlying architecture, which is x86 in my case. So, if I run gcc space, hello dot c, it will create an executable file called a dot out. So, here it is and it contains all the instructions which CPU can understand. So now, if I run a dot out, you can see that it printed, hello world on the output screen.

So, what happens when I run this a dot out? As we know that every running program is a process, so when I run this program, it will create a new process in the RAM. It will copy this a dot out from hard disk to the main memory and create the memory image and then CPU will start executing the instructions. So, is there a way to see all the processes which are there in the RAM? There is a command called top and I will open a new terminal.

(Refer Slide Time: 02:00)



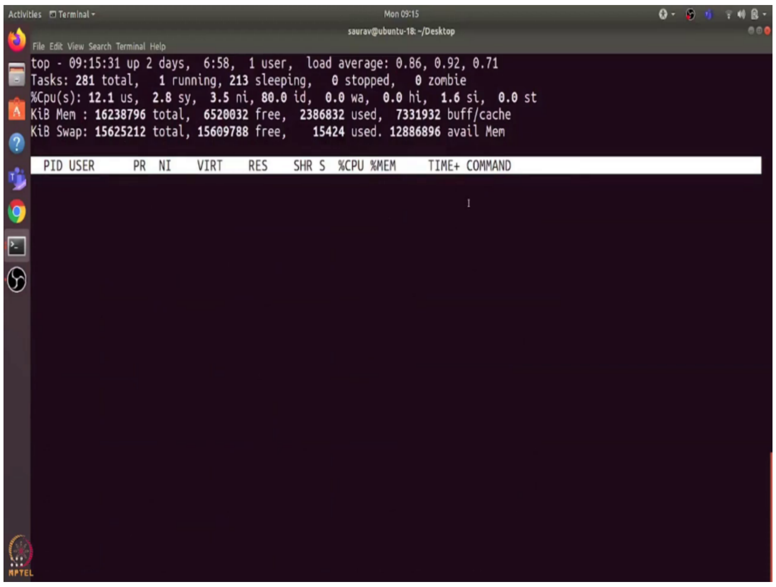
```
saurav@ubuntu-18:~/Desktop$
```



```
top - 09:14:36 up 2 days,  6:57,  1 user,  load average: 1.22, 0.98, 0.72
Tasks: 279 total,   1 running, 212 sleeping,   0 stopped,   0 zombie
%Cpu(s): 19.9 us,  5.7 sy,  0.3 ni, 72.2 id,  0.1 wa,  0.0 hi,  1.8 si,  0.0 st
KiB Mem : 16238796 total,  6538916 free,  2381992 used,  7317888 buff/cache
KiB Swap: 15625212 total, 15609788 free,    15424 used. 12901852 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
14802 saurav    20   0 3503320 365916 131456 S  94.4  2.3   4:28.68 obs
 1056 saurav    20   0 2361880 169320 119504 S  16.7  1.0 201:29.56 Xorg
15103 saurav    20   0   51324   4076   3352 R  16.7  0.0   0:00.05 top
 1308 saurav    20   0 2652516  14792  10440 S  11.1  0.1  81:48.62 pulseaudio
    1 root      20   0  225800   8664   6120 S   5.6  0.1   8:20.65 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.17 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
    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   0:08.32 ksoftirqd/0
   11 root      20   0       0      0      0 I   0.0  0.0   4:35.16 rcu_sched
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.95 migration/0
   13 root     -51   0       0      0      0 S   0.0  0.0   0:00.00 idle_inject/0
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   15 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   16 root     -51   0       0      0      0 S   0.0  0.0   0:00.00 idle_inject/1
   17 root      rt   0       0      0      0 S   0.0  0.0   0:04.42 migration/1
   18 root      20   0       0      0      0 S   0.0  0.0   0:05.12 ksoftirqd/1
   20 root       0 -20       0      0      0 I   0.0  0.0   0:00.00 kworker/1:0H-kb
   21 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/2
   22 root     -51   0       0      0      0 S   0.0  0.0   0:00.00 idle_inject/2
   23 root      rt   0       0      0      0 S   0.0  0.0   0:04.02 migration/2
   24 root      20   0       0      0      0 S   0.0  0.0   0:03.96 ksoftirqd/2
```

So, this top command shows us a list of all the processes and it updates every few seconds. Let us change it to update every one second, so I will use this d argument. So, now it is updating every second and showing me older processes. So, now what I will do is I will run a dot out and try to see a corresponding process in the top output. But there are too many processes, so I will press the O key and filter it based on command equal to a dot out.

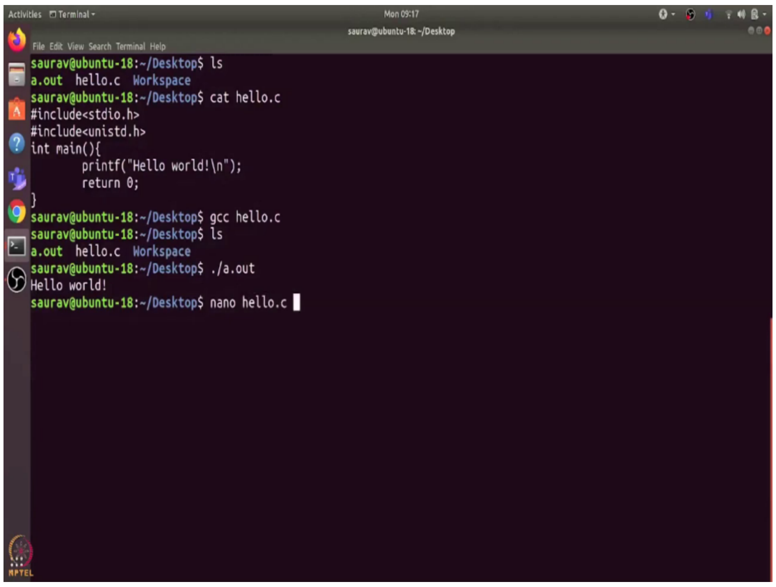
(Refer Slide Time: 02:32)



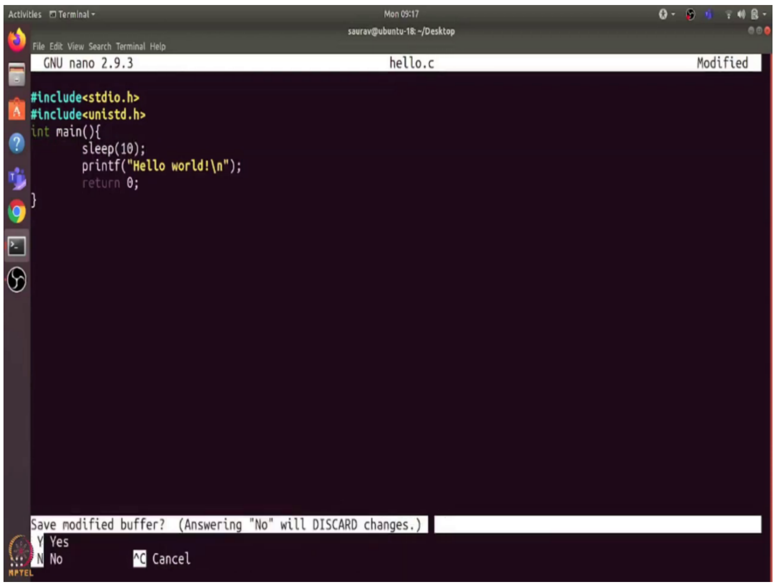
```
top - 09:15:31 up 2 days,  6:58,  1 user,  load average: 0.86, 0.92, 0.71
Tasks: 281 total,   1 running, 213 sleeping,   0 stopped,   0 zombie
%Cpu(s): 12.1 us,  2.8 sy,  3.5 ni, 80.0 id,  0.0 wa,  0.0 hi,  1.6 si,  0.0 st
KiB Mem : 16238796 total,  6520032 free,  2386832 used,  7331932 buff/cache
KiB Swap: 15625212 total, 15609788 free,    15424 used. 12886896 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
```

So, you can see that there is no process currently running with, a dot out as command.

(Refer Slide Time: 02:42)



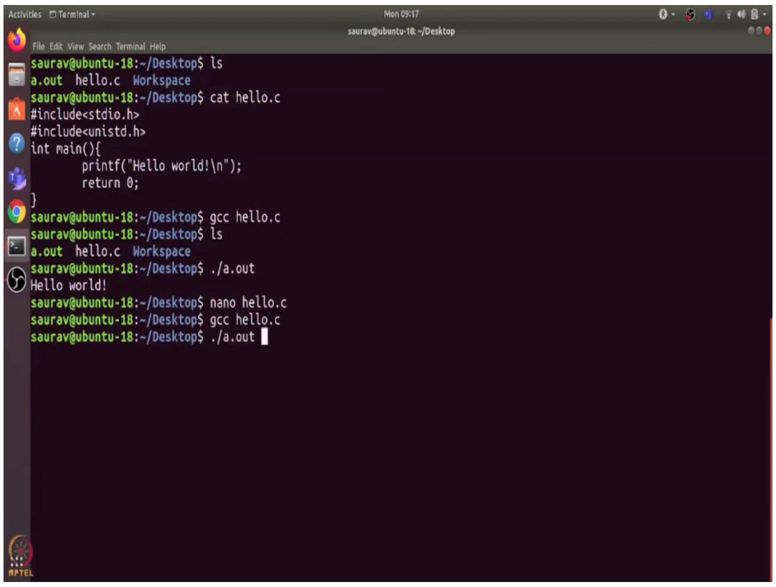
```
saurav@ubuntu-18:~/Desktop$ ls
a.out  hello.c  Workspace
saurav@ubuntu-18:~/Desktop$ cat hello.c
#include<stdio.h>
#include<unistd.h>
int main(){
        printf("Hello world!\n");
        return 0;
}
saurav@ubuntu-18:~/Desktop$ gcc hello.c
saurav@ubuntu-18:~/Desktop$ ls
a.out  hello.c  Workspace
saurav@ubuntu-18:~/Desktop$ ./a.out
Hello world!
saurav@ubuntu-18:~/Desktop$ nano hello.c
```



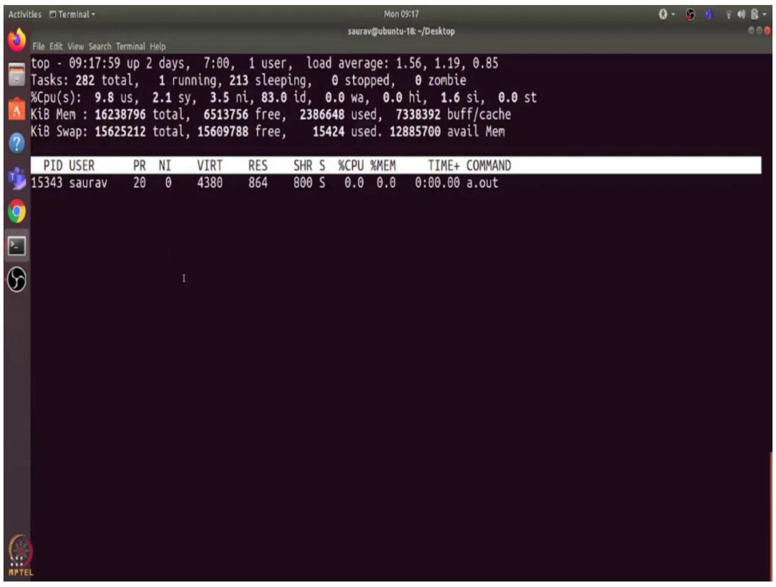
```
GNU nano 2.9.3                     hello.c                          Modified

#include<stdio.h>
#include<unistd.h>
int main(){
        sleep(10);
        printf("Hello world!\n");
        return 0;
}

Save modified buffer?  (Answering "No" will DISCARD changes.)
 Y Yes
 N No           ^C Cancel
```

So, if I just directed an a dot out, it is too fast to see a process popping up in the tops output. So, what I will do is I will add a sleep statement in hello dot c file. So, that it first waits for 10 seconds before executing the printf statement.

(Refer Slide Time: 02:58)



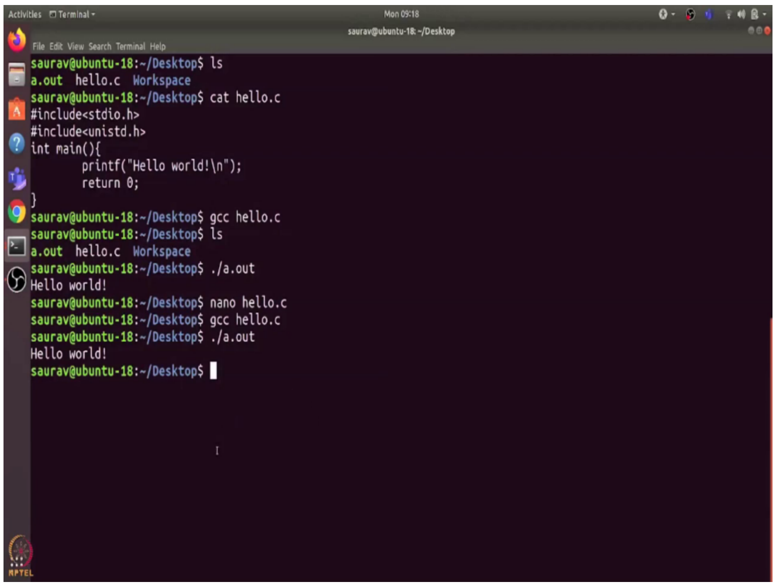
```
saurav@ubuntu-18:~/Desktop$ ls
a.out  hello.c  Workspace
saurav@ubuntu-18:~/Desktop$ cat hello.c
#include<stdio.h>
#include<unistd.h>
int main(){
        printf("Hello world!\n");
        return 0;
}
saurav@ubuntu-18:~/Desktop$ gcc hello.c
saurav@ubuntu-18:~/Desktop$ ls
a.out  hello.c  Workspace
saurav@ubuntu-18:~/Desktop$ ./a.out
Hello world!
saurav@ubuntu-18:~/Desktop$ nano hello.c
saurav@ubuntu-18:~/Desktop$ gcc hello.c
saurav@ubuntu-18:~/Desktop$ ./a.out
```



```
top - 09:17:59 up 2 days,  7:00,  1 user,  load average: 1.56, 1.19, 0.85
Tasks: 282 total,   1 running, 213 sleeping,   0 stopped,   0 zombie
%Cpu(s):  9.8 us,  2.1 sy,  3.5 ni, 83.0 id,  0.0 wa,  0.0 hi,  1.6 si,  0.0 st
KiB Mem : 16238796 total,  6513756 free,  2386648 used,  7338392 buff/cache
KiB Swap: 15625212 total, 15609788 free,    15424 used. 12885700 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
15343 saurav    20   0    4380    864    800 S   0.0  0.0   0:00.00 a.out
```

I will save this file. And now, because we have changed hello dot c, we need to compile it again to update the executable. Now, if I run a dot out and go to the second terminal, we can see that there is a new process with a dot out is command and this is the pid. So, this is the (())(03:15), which is running in CPU. It will wait for 10 seconds and print hello world. And after it prints, it will exit.

(Refer Slide Time: 03:22)



```
saurav@ubuntu-18:~/Desktop$ ls
a.out  hello.c  Workspace
saurav@ubuntu-18:~/Desktop$ cat hello.c
#include<stdio.h>
#include<unistd.h>
int main(){
        printf("Hello world!\n");
        return 0;
}
saurav@ubuntu-18:~/Desktop$ gcc hello.c
saurav@ubuntu-18:~/Desktop$ ls
a.out  hello.c  Workspace
saurav@ubuntu-18:~/Desktop$ ./a.out
Hello world!
saurav@ubuntu-18:~/Desktop$ nano hello.c
saurav@ubuntu-18:~/Desktop$ gcc hello.c
saurav@ubuntu-18:~/Desktop$ ./a.out
Hello world!
saurav@ubuntu-18:~/Desktop$ 
```

So that's the overall life cycle of a C program. We write a C code, we compile it to create an executable, which a CPU can understand. And when we run an executable, it will create a new process in the RAM and create the memory image. The CPU executes all the instructions and then exits. So, that is it for this video. Thanks, and have a nice day.