Operating Systems Lab 2026

Assignment Set 1

Objectives

- Linux installation
- To understand occurrence of race condition in OS
- To learn some useful commands ps, kill, cat, chmod, ls, pwd, man etc.
- To learn the use of some basic system calls getpid(), getppid(), fork()

A1

Question:

Learn the use of ps, kill, cat, chmod, ls, pwd, man etc. commands. Try to be conversant with all the possible options for each command. Try to use man command to know the details of a command. Example: $ man ps

Code:

1. pwd (Print Working Directory)

```
pwd
```

2. ls (List Directory Contents)

```
ls
```

```
ls -l
```

```
ls -a
```

```
ls -la
```

3. cat (Display File Contents)

```
cat file.txt
```

```
cat -n file.txt
```

4. chmod (Change File Permissions)

```
chmod 755 file.txt
```

```
chmod u+x script.sh
```

5. ps (Process Status)

```
ps

ps -ef

ps aux
```

6. kill (Terminate Process)

```
kill PID

kill -9 PID
```

7. man (Manual Pages)

```
man ps

man ls
```

Ouput:

```
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ pwd
/home/subhro/4sem/css452/a1
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ls
'2026 Assignment 1.docx'   a2q3loop.c        a.out
 a1q2bal_update.c          a2q3par-child.c   assign1.docx
 a1q2init_bal.c            acc_balance.txt   assign1os.docx
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ls -l
total 196
-rw-rw-r-- 1 subhro subhro 79330 Jan 14 11:35 '2026 Assignment 1.docx'
-rw-rw-r-- 1 subhro subhro   311 Jan 14 11:42  a1q2bal_update.c
-rw-rw-r-- 1 subhro subhro   184 Jan 14 11:41  a1q2init_bal.c
-rw-rw-r-- 1 subhro subhro   122 Jan 14 11:58  a2q3loop.c
-rw-rw-r-- 1 subhro subhro   537 Jan 14 11:44  a2q3par-child.c
-rw-rw-r-- 1 subhro subhro     5 Jan 14 11:43  acc_balance.txt
-rwxrwxr-x 1 subhro subhro 16008 Jan 14 11:59  a.out
-rw-rw-r-- 1 subhro subhro 37517 Jan 15 13:50  assign1.docx
-rw-rw-r-- 1 subhro subhro 37600 Jan 15 13:53  assign1os.docx
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ls -a
 .                         a1q2init_bal.c    a.out
 ..                        a2q3loop.c        assign1.docx
'2026 Assignment 1.docx'   a2q3par-child.c   assign1os.docx
 a1q2bal_update.c          acc_balance.txt   .~lock.assign1os.docx#
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ls -la
total 208
drwxrwxr-x 2 subhro subhro  4096 Jan 15 13:54  .
drwxrwxr-x 3 subhro subhro  4096 Jan 14 11:32  ..
-rw-rw-r-- 1 subhro subhro 79330 Jan 14 11:35 '2026 Assignment 1.docx'
-rw-rw-r-- 1 subhro subhro   311 Jan 14 11:42  a1q2bal_update.c
-rw-rw-r-- 1 subhro subhro   184 Jan 14 11:41  a1q2init_bal.c
-rw-rw-r-- 1 subhro subhro   122 Jan 14 11:58  a2q3loop.c
-rw-rw-r-- 1 subhro subhro   537 Jan 14 11:44  a2q3par-child.c
-rw-rw-r-- 1 subhro subhro     5 Jan 14 11:43  acc_balance.txt
-rwxrwxr-x 1 subhro subhro 16008 Jan 14 11:59  a.out
-rw-rw-r-- 1 subhro subhro 37517 Jan 15 13:50  assign1.docx
-rw-rw-r-- 1 subhro subhro 37600 Jan 15 13:53  assign1os.docx
-rw-rw-r-- 1 subhro subhro   112 Jan 15 13:54  .~lock.assign1os.docx#
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ cat a1q2bal_update.c
#include <stdio.h>
#include <unistd.h>

int main() {
```

```
int main() {
    FILE *fp;
    int bal;

    fp = fopen("acc_balance.txt", "r");
    fscanf(fp, "%d", &bal);
    fclose(fp);

    bal = bal - 50;
    usleep(100000);

    fp = fopen("acc_balance.txt", "w");
    fprintf(fp, "%d\n", bal);
    fclose(fp);

    return 0;
}

subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ cat -n a1q2bal_update.c
     1  #include <stdio.h>
     2  #include <unistd.h>
     3
     4  int main() {
     5      FILE *fp;
     6      int bal;
     7
     8      fp = fopen("acc_balance.txt", "r");
     9      fscanf(fp, "%d", &bal);
    10      fclose(fp);
    11
    12      bal = bal - 50;
    13      usleep(100000);
    14
    15      fp = fopen("acc_balance.txt", "w");
    16      fprintf(fp, "%d\n", bal);
    17      fclose(fp);
    18
    19      return 0;
    20  }
    21
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ gedit test.c
```

```
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ gedit checklist.c
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ chmod 755 chechklist.c
chmod: cannot access 'chechklist.c': No such file or directory
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ chmod 755 checklist.c
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ chmod u+x checklist.c
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ps
    PID TTY          TIME CMD
   8673 pts/0    00:00:00 bash
   8937 pts/0    00:00:00 ps
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 13:22 ?        00:00:03 /sbin/init splash
root           2       0  0 13:22 ?        00:00:00 [kthreadd]
root           3       2  0 13:22 ?        00:00:00 [pool_workqueue_release]
root           4       2  0 13:22 ?        00:00:00 [kworker/R-rcu_gp]
root           5       2  0 13:22 ?        00:00:00 [kworker/R-sync_wq]
root           6       2  0 13:22 ?        00:00:00 [kworker/R-kvfree_rcu_reclai
root           7       2  0 13:22 ?        00:00:00 [kworker/R-slub_flushwq]
root           8       2  0 13:22 ?        00:00:00 [kworker/R-netns]
root          10       2  0 13:22 ?        00:00:00 [kworker/0:1-events]
root          11       2  0 13:22 ?        00:00:00 [kworker/0:0H-events_highpri
root          13       2  0 13:22 ?        00:00:00 [kworker/R-mm_percpu_wq]
root          14       2  0 13:22 ?        00:00:00 [rcu_tasks_kthread]
root          15       2  0 13:22 ?        00:00:00 [rcu_tasks_rude_kthread]
root          16       2  0 13:22 ?        00:00:00 [rcu_tasks_trace_kthread]
root          17       2  0 13:22 ?        00:00:00 [ksoftirqd/0]
root          18       2  0 13:22 ?        00:00:05 [rcu_preempt]
root          19       2  0 13:22 ?        00:00:00 [rcu_exp_par_gp_kthread_work
root          20       2  0 13:22 ?        00:00:00 [rcu_exp_gp_kthread_worker]
root          21       2  0 13:22 ?        00:00:00 [migration/0]
root          22       2  0 13:22 ?        00:00:00 [idle_inject/0]
root          23       2  0 13:22 ?        00:00:00 [cpuhp/0]
root          24       2  0 13:22 ?        00:00:00 [cpuhp/2]
root          25       2  0 13:22 ?        00:00:00 [idle_inject/2]
root          26       2  0 13:22 ?        00:00:00 [migration/2]
root          27       2  0 13:22 ?        00:00:00 [ksoftirqd/2]
root          29       2  0 13:22 ?        00:00:00 [kworker/2:0H-events_highpri
root          30       2  0 13:22 ?        00:00:00 [cpuhp/4]
root          31       2  0 13:22 ?        00:00:00 [idle_inject/4]
root          32       2  0 13:22 ?        00:00:00 [migration/4]
root          33       2  0 13:22 ?        00:00:00 [ksoftirqd/4]
```

```
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.0  23560 14668 ?        Ss   13:22   0:03 /sbin/init sp
root           2  0.0  0.0      0     0 ?        S    13:22   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    13:22   0:00 [pool_workque
root           4  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/R-rc
root           5  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/R-sy
root           6  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/R-kv
root           7  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/R-sl
root           8  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/R-ne
root          10  0.0  0.0      0     0 ?        I    13:22   0:00 [kworker/0:1-
root          11  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/0:0H
root          13  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/R-mm
root          14  0.0  0.0      0     0 ?        I    13:22   0:00 [rcu_tasks_kt
root          15  0.0  0.0      0     0 ?        I    13:22   0:00 [rcu_tasks_ru
root          16  0.0  0.0      0     0 ?        I    13:22   0:00 [rcu_tasks_tr
root          17  0.0  0.0      0     0 ?        S    13:22   0:00 [ksoftirqd/0]
root          18  0.1  0.0      0     0 ?        I    13:22   0:05 [rcu_preempt]
root          19  0.0  0.0      0     0 ?        S    13:22   0:00 [rcu_exp_par_
root          20  0.0  0.0      0     0 ?        S    13:22   0:00 [rcu_exp_gp_k
root          21  0.0  0.0      0     0 ?        S    13:22   0:00 [migration/0]
root          22  0.0  0.0      0     0 ?        S    13:22   0:00 [idle_inject/
root          23  0.0  0.0      0     0 ?        S    13:22   0:00 [cpuhp/0]
root          24  0.0  0.0      0     0 ?        S    13:22   0:00 [cpuhp/2]
root          25  0.0  0.0      0     0 ?        S    13:22   0:00 [idle_inject/
root          26  0.0  0.0      0     0 ?        S    13:22   0:00 [migration/2]
root          27  0.0  0.0      0     0 ?        S    13:22   0:00 [ksoftirqd/2]
root          29  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/2:0H
root          30  0.0  0.0      0     0 ?        S    13:22   0:00 [cpuhp/4]
root          31  0.0  0.0      0     0 ?        S    13:22   0:00 [idle_inject/
root          32  0.0  0.0      0     0 ?        S    13:22   0:00 [migration/4]
root          33  0.0  0.0      0     0 ?        S    13:22   0:00 [ksoftirqd/4]
root          35  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/4:0H
root          36  0.0  0.0      0     0 ?        S    13:22   0:00 [cpuhp/6]
root          37  0.0  0.0      0     0 ?        S    13:22   0:00 [idle_inject/
root          38  0.0  0.0      0     0 ?        S    13:22   0:00 [migration/6]
root          39  0.0  0.0      0     0 ?        S    13:22   0:00 [ksoftirqd/6]
root          41  0.0  0.0      0     0 ?        I<   13:22   0:00 [kworker/6:0H
root          42  0.0  0.0      0     0 ?        S    13:22   0:00 [cpuhp/8]
root          43  0.0  0.0      0     0 ?        S    13:22   0:00 [idle_inject/
root          44  0.0  0.0      0     0 ?        S    13:22   0:00 [migration/8]
root          45  0.0  0.0      0     0 ?        S    13:22   0:00 [ksoftirqd/8]
```

```
subhro      3716  0.0  0.4 1459917592 75468 ?    Sl   13:24   0:00 /opt/brave.co
subhro      3763  0.0  0.2 567508 33956 ?        Sl   13:25   0:00 /usr/bin/upda
subhro      3835  0.0  0.0 388228  8752 ?        Sl   13:25   0:00 /usr/libexec/
subhro      3944  0.2  1.2 1626884 200576 ?      Sl   13:28   0:05 /usr/bin/naut
root        3963  0.0  0.0      0     0 ?        S    13:28   0:00 [nvidia-drm/t
root        4585  0.0  0.0      0     0 ?        I    13:30   0:00 [kworker/14:0
root        4677  0.0  0.0      0     0 ?        I<   13:30   0:00 [kworker/R-tl
root        5455  0.0  0.0      0     0 ?        I    13:31   0:00 [kworker/2:0-
root        5595  0.0  0.0      0     0 ?        S    13:31   0:00 [psimon]
root        6184  0.0  0.0      0     0 ?        I<   13:34   0:00 [kworker/u65:
root        6196  0.0  0.0      0     0 ?        I    13:37   0:00 [kworker/14:1
root        8064  0.1  0.2 603652 46180 ?        Ssl  13:43   0:01 /usr/libexec/
root        8072  0.0  0.0      0     0 ?        I    13:43   0:00 [kworker/6:1-
root        8111  0.0  0.0      0     0 ?        I    13:44   0:01 [kworker/u64:
subhro      8210  0.0  0.0 162512  6704 ?        Sl   13:48   0:00 /usr/lib/libr
subhro      8251  4.8  3.1 8639884 503744 ?      Sl   13:48   1:03 /usr/lib/libr
root        8358  0.0  0.0      0     0 ?        I    13:50   0:00 [kworker/0:0-
root        8367  0.0  0.0      0     0 ?        S    13:50   0:00 [nvidia-drm/t
subhro      8375  0.0  0.0 462336  8980 ?        Sl   13:50   0:00 /usr/libexec/
subhro      8385  0.0  0.0 390548  9160 ?        Sl   13:50   0:00 /usr/libexec/
root        8529  0.0  0.0      0     0 ?        I<   13:52   0:00 [kworker/u65:
root        8614  0.1  0.0      0     0 ?        I    13:54   0:01 [kworker/u64:
subhro      8661  0.9  0.3 846348 59128 ?        Ssl  13:57   0:07 /usr/libexec/
subhro      8673  0.0  0.0  11024  5232 pts/0    Ss   13:57   0:00 bash
root        8685  0.0  0.0      0     0 ?        I    13:59   0:00 [kworker/7:0]
root        8686  0.0  0.0      0     0 ?        I    13:59   0:00 [kworker/4:0]
root        8692  0.0  0.0      0     0 ?        I    14:00   0:00 [kworker/10:3
root        8695  0.0  0.0      0     0 ?        I    14:00   0:00 [kworker/11:2
root        8713  0.0  0.0      0     0 ?        I    14:01   0:00 [kworker/u64:
root        8871  0.0  0.0      0     0 ?        I<   14:07   0:00 [kworker/u65:
root        8877  0.1  0.0      0     0 ?        I    14:07   0:00 [kworker/u64:
root        8879  0.0  0.0      0     0 ?        I    14:08   0:00 [kworker/8:2-
root        8927  0.0  0.0      0     0 ?        I    14:09   0:00 [kworker/6:0-
subhro      8939  0.0  0.0  13744  4736 pts/0    R+   14:10   0:00 ps aux
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ps
    PID TTY          TIME CMD
   8673 pts/0    00:00:00 bash
   9360 pts/0    00:00:00 ps
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ kill 9335
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ man ps
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$
```

A2

Question:

Execute the balance update code discussed in the class and have a feel of race condition in OS. Create the acc_balance.txt file by executing the init_balance process. Create multiple bal_update processes and let them run simultaneously in background. Show the anomalous outcome using cat and provide terminal snapshots.

Code:

Race Condition Demonstration

File name: a1q2init.c

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    FILE *fp = fopen("acc_balance.txt", "w");
    if (fp == NULL) {
        perror("File open error");
        exit(1);
    }

    fprintf(fp, "%d\n", 1000);
    fclose(fp);

    printf("Account initialized with balance = 1000\n");
    return 0;
}
```

File name: a1q2bal.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    FILE *fp;
    int balance;
```

```
    fp = fopen("acc_balance.txt", "r");
    if (fp == NULL) {
        perror("File open error");
        exit(1);
    }

    fscanf(fp, "%d", &balance);
    fclose(fp);

    balance = balance + 100;

    sleep(1);

    fp = fopen("acc_balance.txt", "w");
    if (fp == NULL) {
        perror("File open error");
        exit(1);
    }

    fprintf(fp, "%d\n", balance);
    fclose(fp);

    printf("Updated balance: %d\n", balance);
    return 0;
}
```

Output :

```
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ gedit a1q2init.c
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ gedit a1q2bal.c
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ gcc a1q2init.c
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ./a.out
Account initialized with balance = 1000
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ gcc a1q2bal.c
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ./a.out
Updated balance: 1100
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ./a.out
Updated balance: 1200
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$
```

A3

Question:

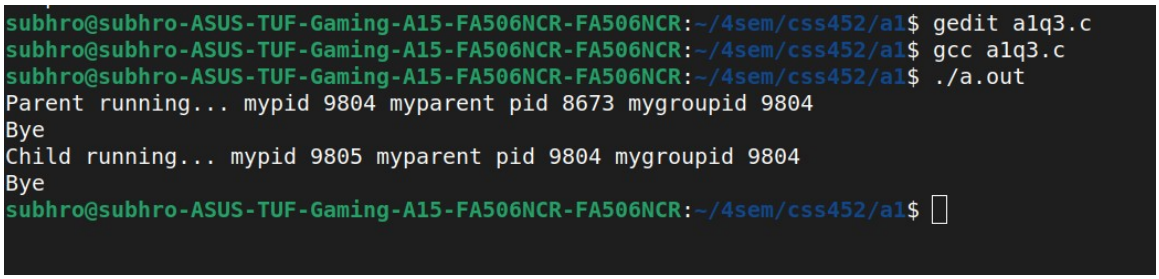
Write a code where the parent process creates a child process using fork(). The child and parent should print their PIDs, parent PID, group ID, and print 'Bye' before termination.

Code:

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <stdlib.h>


int main() {

  pid_t pid;


  pid = fork();


  if (pid < 0) {

    perror("Fork failed");

    exit(1);

  }


  if (pid == 0) {

    printf("Child running... mypid %d myparent pid %d mygroupid %d\n",

        getpid(), getppid(), getpgrp());

    printf("Bye\n");

  } else {

    printf("Parent running... mypid %d myparent pid %d mygroupid %d\n",
```

```
            getpid(), getppid(), getpgrp());
        printf("Bye\n");
    }

    return 0;
}
```

Output :



A4

Question:

Write one process executing an infinite loop and another process infinitely calling malloc(). Run both in background. Use ps command to observe CPU and memory usage. Provide at least three snapshots.

Code:

File name: `a1q4_cpu.c`
(Infinite loop – CPU intensive)

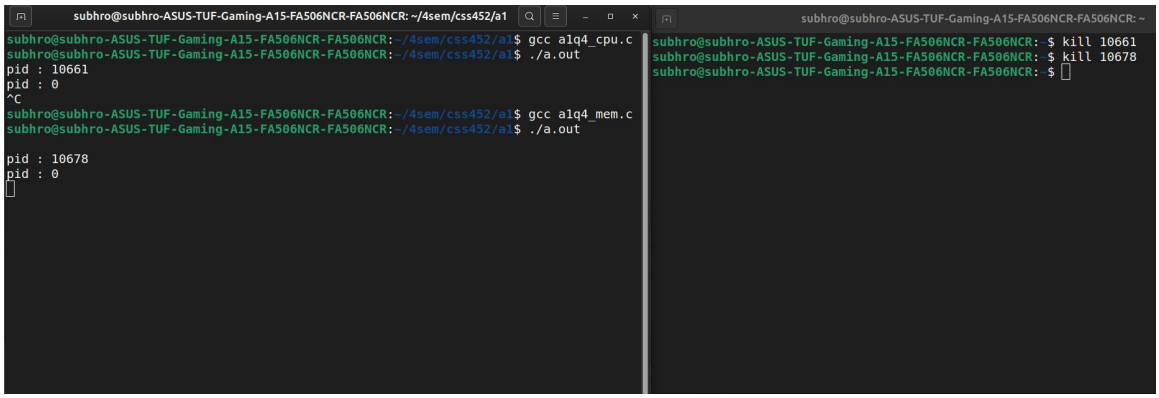
```
#include <stdio.h>


int main() {
    int pid = fork();
      printf("pid : %d\n",pid);
    while (1) {
    }
    return 0;
}
```

File name: `a1q4_mem.c`
(Infinite malloc – Memory intensive)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


int main() {
    int pid = fork();
      printf("pid : %d\n",pid);
    while (1) {
```

```
        malloc(1024 * 1024);
        sleep(1);
    }
    return 0;
}
```

Output :



A5

Question:

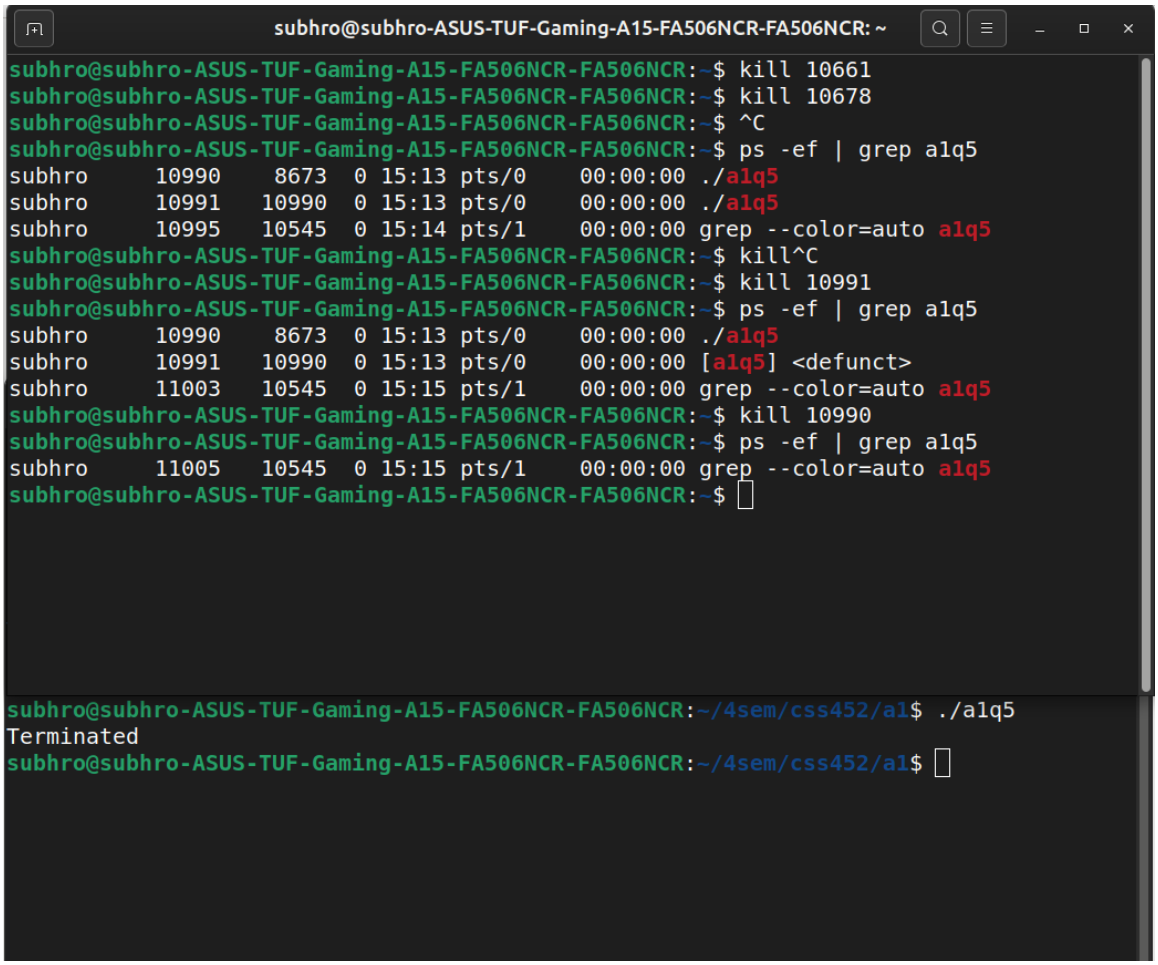
Write a code where parent creates a child process and both run infinite loops. Terminate child to show zombie process and terminate parent to show orphan process using ps snapshots.

Code:

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <stdlib.h>


int main() {

  pid_t pid;


  pid = fork();


  if (pid < 0) {

    perror("Fork failed");

    exit(1);

  }


  if (pid == 0) {

    while (1) {

      sleep(1);

    }

  } else {

    while (1) {
```

```
        sleep(1);
      }
    }

    return 0;
}
```

Snapshots:



A6

Question:

Write a utility program that executes ps aux and ps -ef using system() call, redirects output to files, parses them and reports process statistics.

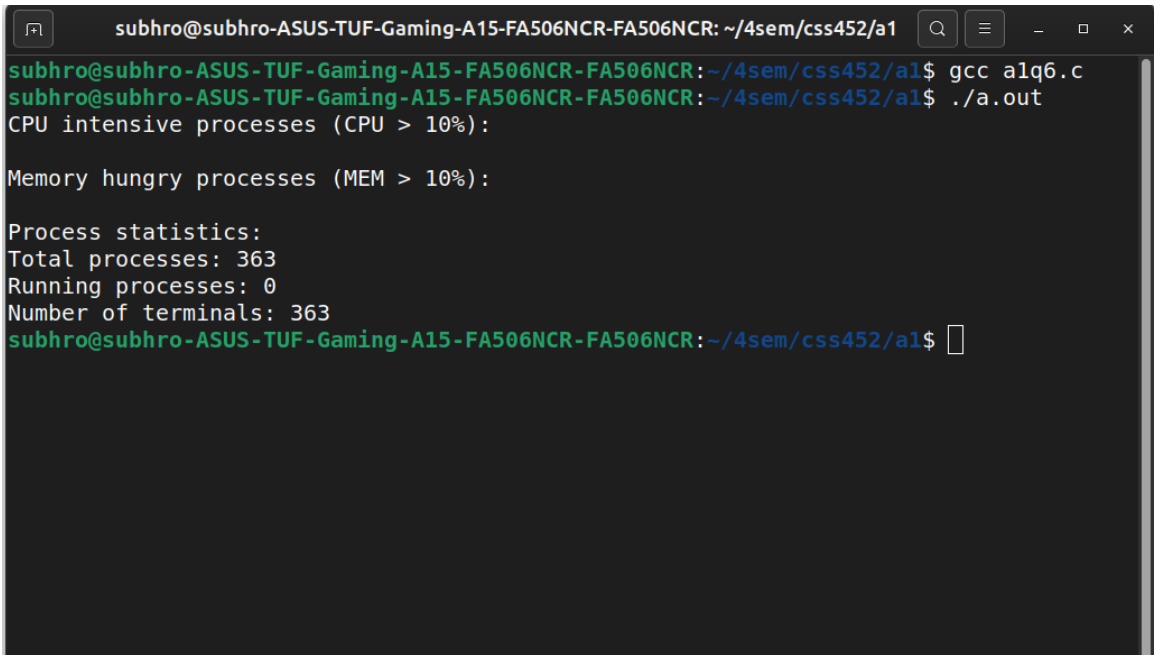
Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


int main() {

   FILE *fp;

   char line[1024];

   int total = 0, running = 0, terminals = 0;


   system("ps aux > ps_aux.txt");

   system("ps -ef > ps_ef.txt");


   fp = fopen("ps_aux.txt", "r");

   if (fp == NULL) {

      perror("File open error");

      exit(1);

   }


   fgets(line, sizeof(line), fp);


   printf("CPU intensive processes (CPU > 10%%):\n");
```

```
while (fgets(line, sizeof(line), fp)) {
    total++;

    char user[50], tty[50];
    int pid;
    float cpu, mem;

    sscanf(line, "%s %d %f %f %*s %*s %*s %*s %s",
           user, &pid, &cpu, &mem, tty);

    if (cpu > 10.0)
        printf("PID: %d CPU: %.2f\n", pid, cpu);

    if (strcmp(tty, "?") != 0)
        terminals++;

    if (strstr(line, " R "))
        running++;
}
fclose(fp);

fp = fopen("ps_ef.txt", "r");
if (fp == NULL) {
    perror("File open error");
    exit(1);
```

```
    }

    fgets(line, sizeof(line), fp);

    printf("\nMemory hungry processes (MEM > 10%%):\n");
    while (fgets(line, sizeof(line), fp)) {
        int pid;
        float mem;

        if (sscanf(line, "%*s %d %*s %*s %*s %*s %f",
                   &pid, &mem) == 2) {
            if (mem > 10.0)
                printf("PID: %d MEM: %.2f\n", pid, mem);
        }
    }
    fclose(fp);

    printf("\nProcess statistics:\n");
    printf("Total processes: %d\n", total);
    printf("Running processes: %d\n", running);
    printf("Number of terminals: %d\n", terminals);

    return 0;
}
```

Snapshots:


subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR: ~/4sem/css452/a1
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ gcc a1q6.c
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ ./a.out
CPU intensive processes (CPU > 10%):
Memory hungry processes (MEM > 10%):
Process statistics:
Total processes: 363
Running processes: 0
Number of terminals: 363
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$

A7

Question:

Run lscpu command and find number of CPU cores. Use man lscpu to understand all displayed information.

Code:

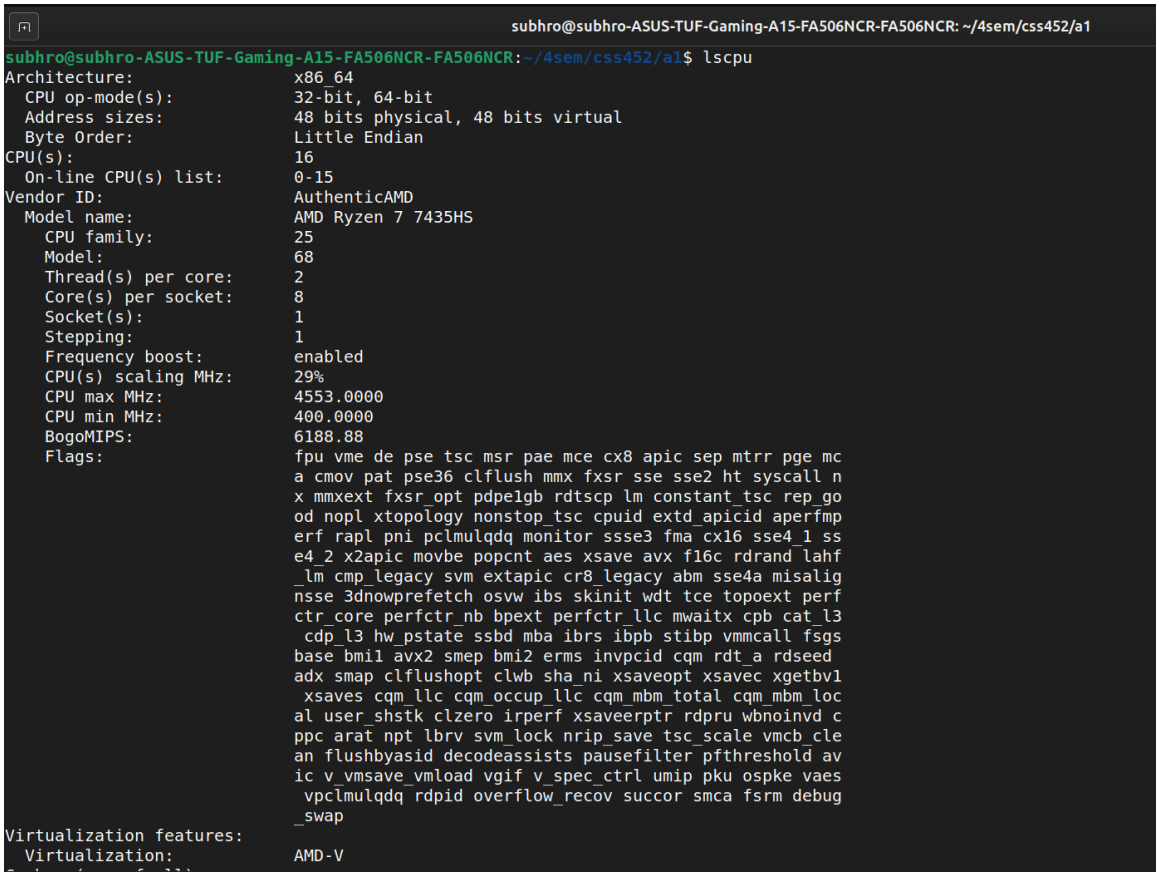
Command to display CPU details

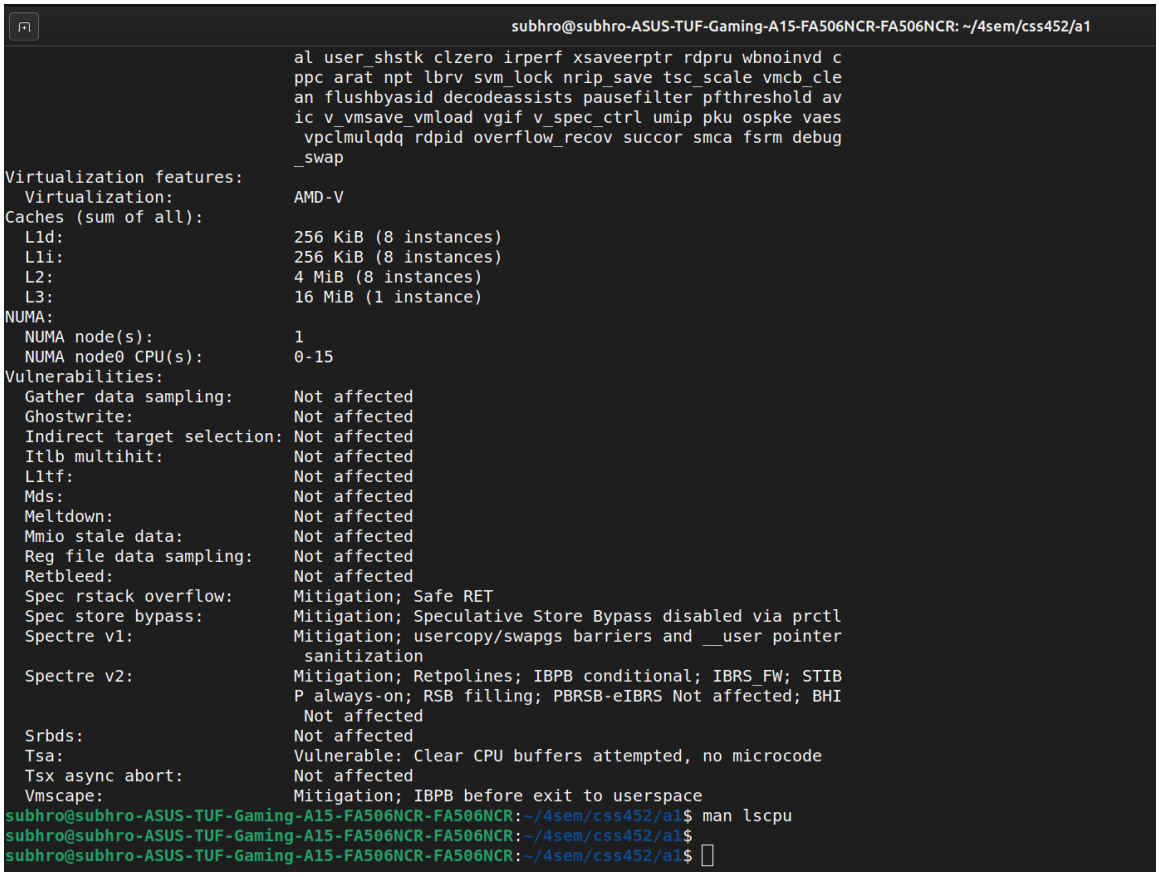
```
lscpu
```

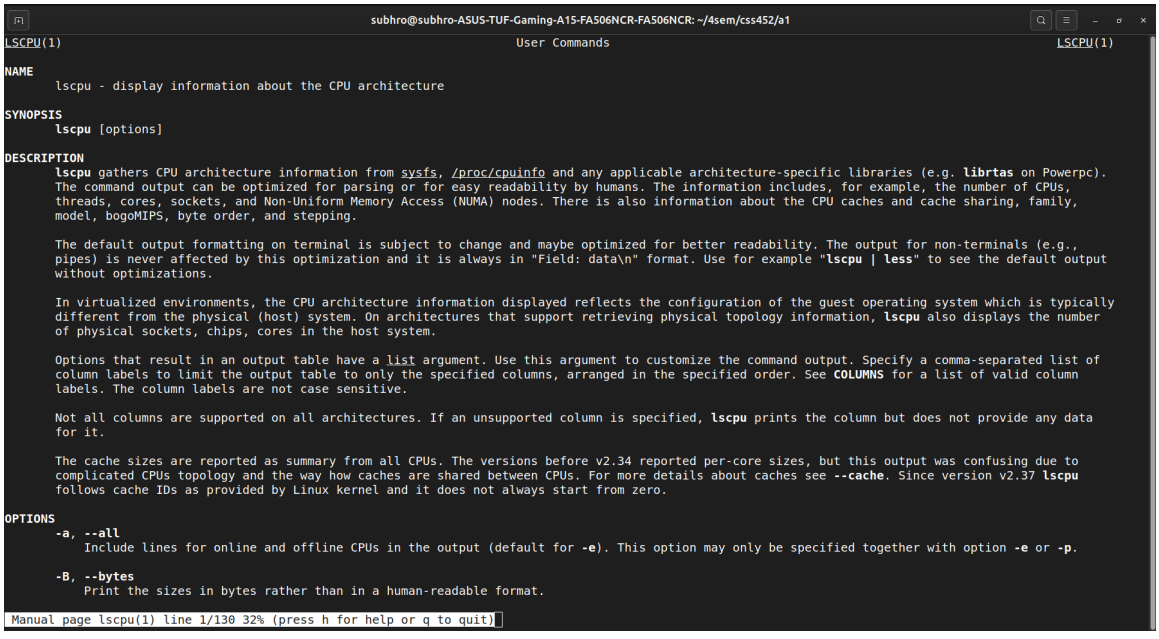
Command to understand lscpu output

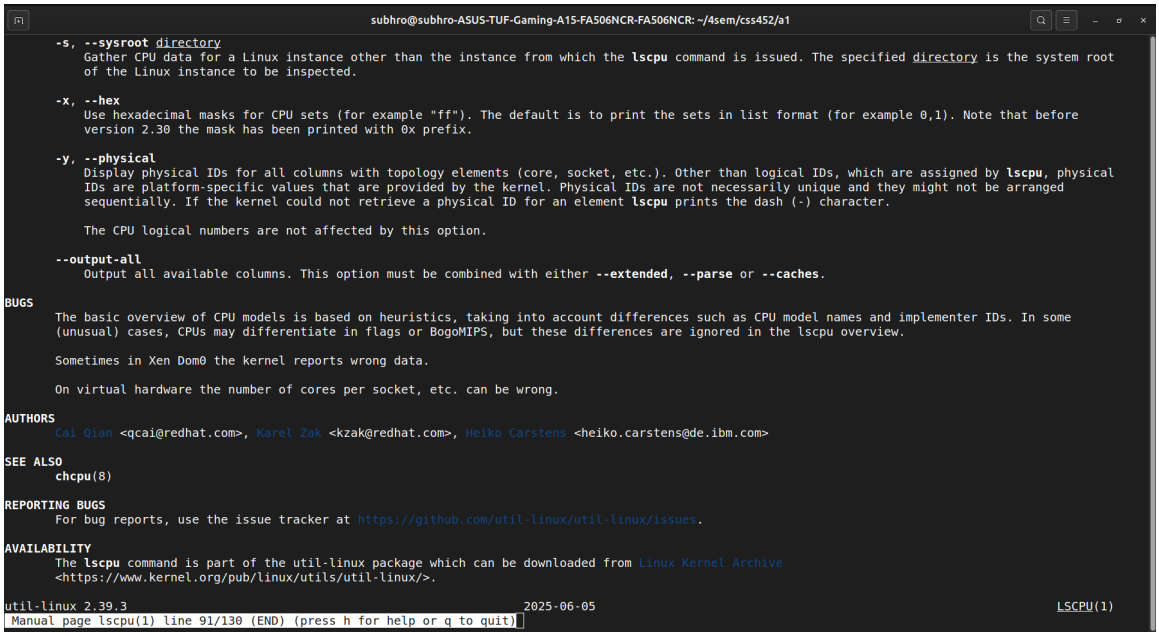
```
man lscpu
```

Output :




subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR: ~/4sem/css452/a1
al user_shstk clzero irperf xsaveerptr rdpru wbnoinvd c
ppc arat npt lbrv svm_lock nrip_save tsc_scale vmcb_cle
an flushbyasid decodeassists pausefilter pfthreshold av
ic v_vmsave_vmload vgif v_spec_ctrl umip pku ospke vaes
vpclmulqdq rdpid overflow_recov succor smca fsrm debug
_swap
Virtualization features:
Virtualization: AMD-V
Caches (sum of all):
L1d: 256 KiB (8 instances)
L1i: 256 KiB (8 instances)
L2: 4 MiB (8 instances)
L3: 16 MiB (1 instance)
NUMA:
NUMA node(s): 1
NUMA node0 CPU(s): 0-15
Vulnerabilities:
Gather data sampling: Not affected
Ghostwrite: Not affected
Indirect target selection: Not affected
Itlb multihit: Not affected
L1tf: Not affected
Mds: Not affected
Meltdown: Not affected
Mmio stale data: Not affected
Reg file data sampling: Not affected
Retbleed: Not affected
Spec rstack overflow: Mitigation; Safe RET
Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl
Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer
sanitization
Spectre v2: Mitigation; Retpolines; IBPB conditional; IBRS_FW; STIB
P always-on; RSB filling; PBRSB-eIBRS Not affected; BHI
Not affected
Srbds: Not affected
Tsa: Vulnerable: Clear CPU buffers attempted, no microcode
Tsx async abort: Not affected
Vmscape: Mitigation; IBPB before exit to userspace
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$ man lscpu
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$
subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR:~/4sem/css452/a1$


subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR: ~/4sem/css452/a1
LSCPU(1) User Commands LSCPU(1)
NAME
lscpu - display information about the CPU architecture
SYNOPSIS
lscpu [options]
DESCRIPTION
lscpu gathers CPU architecture information from sysfs, /proc/cpuinfo and any applicable architecture-specific libraries (e.g. librtas on Powerpc). The command output can be optimized for parsing or for easy readability by humans. The information includes, for example, the number of CPUs, threads, cores, sockets, and Non-Uniform Memory Access (NUMA) nodes. There is also information about the CPU caches and cache sharing, family, model, bogoMIPS, byte order, and stepping.
The default output formatting on terminal is subject to change and maybe optimized for better readability. The output for non-terminals (e.g., pipes) is never affected by this optimization and it is always in "Field: data\n" format. Use for example "lscpu | less" to see the default output without optimizations.
In virtualized environments, the CPU architecture information displayed reflects the configuration of the guest operating system which is typically different from the physical (host) system. On architectures that support retrieving physical topology information, lscpu also displays the number of physical sockets, chips, cores in the host system.
Options that result in an output table have a list argument. Use this argument to customize the command output. Specify a comma-separated list of column labels to limit the output table to only the specified columns, arranged in the specified order. See COLUMNS for a list of valid column labels. The column labels are not case sensitive.
Not all columns are supported on all architectures. If an unsupported column is specified, lscpu prints the column but does not provide any data for it.
The cache sizes are reported as summary from all CPUs. The versions before v2.34 reported per-core sizes, but this output was confusing due to complicated CPUs topology and the way how caches are shared between CPUs. For more details about caches see --cache. Since version v2.37 lscpu follows cache IDs as provided by Linux kernel and it does not always start from zero.
OPTIONS
-a, --all
Include lines for online and offline CPUs in the output (default for -e). This option may only be specified together with option -e or -p.
-B, --bytes
Print the sizes in bytes rather than in a human-readable format.
Manual page lscpu(1) line 1/130 32% (press h for help or q to quit)


subhro@subhro-ASUS-TUF-Gaming-A15-FA506NCR-FA506NCR: ~/4sem/css452/a1
-s, --sysroot directory
Gather CPU data for a Linux instance other than the instance from which the lscpu command is issued. The specified directory is the system root of the Linux instance to be inspected.
-x, --hex
Use hexadecimal masks for CPU sets (for example "ff"). The default is to print the sets in list format (for example 0,1). Note that before version 2.30 the mask has been printed with 0x prefix.
-y, --physical
Display physical IDs for all columns with topology elements (core, socket, etc.). Other than logical IDs, which are assigned by lscpu, physical IDs are platform-specific values that are provided by the kernel. Physical IDs are not necessarily unique and they might not be arranged sequentially. If the kernel could not retrieve a physical ID for an element lscpu prints the dash (-) character.
The CPU logical numbers are not affected by this option.
--output-all
Output all available columns. This option must be combined with either --extended, --parse or --caches.
BUGS
The basic overview of CPU models is based on heuristics, taking into account differences such as CPU model names and implementer IDs. In some (unusual) cases, CPUs may differentiate in flags or BogoMIPS, but these differences are ignored in the lscpu overview.
Sometimes in Xen Dom0 the kernel reports wrong data.
On virtual hardware the number of cores per socket, etc. can be wrong.
AUTHORS
Cai Qian <qcai@redhat.com>, Karel Zak <kzak@redhat.com>, Heiko Carstens <heiko.carstens@de.ibm.com>
SEE ALSO
chcpu(8)
REPORTING BUGS
For bug reports, use the issue tracker at https://github.com/util-linux/util-linux/issues.
AVAILABILITY
The lscpu command is part of the util-linux package which can be downloaded from Linux Kernel Archive <https://www.kernel.org/pub/linux/utils/util-linux/>.
util-linux 2.39.3
2025-06-05
LSCPU(1)
Manual page lscpu(1) line 91/130 (END) (press h for help or q to quit)