1. **Process and process IDs.**

   (a) Output:

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  4548 26809  0  80   0 -  1943 wait   pts/26   00:00:00 bash
0 S  1000  4563  4548  0  80   0 -  1944 wait   pts/26   00:00:00 bash
0 S  1000  4580  4563  0  80   0 -  1944 wait   pts/26   00:00:00 bash
0 S  1000 13153  2247  0  80   0 -  2142 wait   pts/26   00:00:01 bash
0 T  1000 16121 13153  0  80   0 -  1074 signal pts/26   00:00:00 rm
1 S  1000 21326  4580  0  80   0 -  1944 wait   pts/26   00:00:00 bash
0 R  1000 21330 21326  0  80   0 -  1248 -      pts/26   00:00:00 ps
0 S  1000 21331 21326  0  80   0 -  1059 pipe_w pts/26   00:00:00 tee
0 S  1000 26809 13153  0  80   0 -  2138 wait   pts/26   00:00:00 bash
0 T  1000 27762 26809  0  80   0 -  3163 signal pts/26   00:00:00 grep
0 T  1000 27837 26809  0  80   0 -  3163 signal pts/26   00:00:00 grep
1 T  1000 28430 26809  0  80   0 -  1951 signal pts/26   00:00:00 bash
1 T  1000 29772 26809  0  80   0 -  1968 signal pts/26   00:00:00 bash
0 T  1000 29773 29772  0  80   0 -  4796 signal pts/26   00:00:00 command-not-fou
0 T  1000 31895 13153  0  80   0 - 27304 signal pts/26   00:00:00 pdflatex
0 T  1000 31900 13153  0  80   0 - 27304 signal pts/26   00:00:00 pdflatex
```

   (b) Zombie:

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  4548 26809  0  80   0 -  1943 wait   pts/26   00:00:00 bash
0 S  1000  4563  4548  0  80   0 -  1944 wait   pts/26   00:00:00 bash
0 S  1000  4580  4563  0  80   0 -  1949 wait   pts/26   00:00:00 bash
0 S  1000 13153  2247  0  80   0 -  2142 wait   pts/26   00:00:01 bash
0 T  1000 16121 13153  0  80   0 -  1074 signal pts/26   00:00:00 rm
1 T  1000 21326  4580  0  80   0 -  1944 signal pts/26   00:00:00 bash
0 T  1000 21375 21326  0  80   0 -  3806 signal pts/26   00:00:00 trix750.elf
0 T  1000 21502  4580  0  80   0 - 27304 signal pts/26   00:00:00 pdflatex
0 T  1000 21533  4580  0  80   0 - 27304 signal pts/26   00:00:00 pdflatex
0 T  1000 21638  4580  0  80   0 - 27304 signal pts/26   00:00:00 pdflatex
0 T  1000 23169  4580  0  80   0 -   507 signal pts/26   00:00:00 zombie.elf
1 Z  1000 23170 23169  0  80   0 -     0 exit   pts/26   00:00:00 zombie.elf <defunct>
0 R  1000 23173  4580  0  80   0 -  1248 -      pts/26   00:00:00 ps
0 S  1000 26809 13153  0  80   0 -  2138 wait   pts/26   00:00:00 bash
0 T  1000 27762 26809  0  80   0 -  3163 signal pts/26   00:00:00 grep
0 T  1000 27837 26809  0  80   0 -  3163 signal pts/26   00:00:00 grep
1 T  1000 28430 26809  0  80   0 -  1951 signal pts/26   00:00:00 bash
1 T  1000 29772 26809  0  80   0 -  1968 signal pts/26   00:00:00 bash
0 T  1000 29773 29772  0  80   0 -  4796 signal pts/26   00:00:00 command-not-fou
0 T  1000 31895 13153  0  80   0 - 27304 signal pts/26   00:00:00 pdflatex
0 T  1000 31900 13153  0  80   0 - 27304 signal pts/26   00:00:00 pdflatex
```

   The parent process is the PID 23169. This is the process that called the child process; and is ultimately its parent process.

   (c) PID less than 10:

```
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 -  1149 poll_s ?        00:00:04 init
1 S     0     2     0  0  80   0 -     0 kthrea ?        00:00:00 kthreadd
1 S     0     3     2  0  80   0 -     0 smpboo ?        00:01:05 ksoftirqd/0
1 S     0     5     2  0  60 -20 -     0 worker ?        00:00:00 kworker/0:0H
1 S     0     7     2  0  80   0 -     0 rcu_gp ?        00:39:53 rcu_sched
1 S     0     8     2  0  80   0 -     0 rcu_gp ?        00:00:00 rcu_bh
1 S     0     9     2  0 -40   - -     0 smpboo ?        00:00:00 migration/0
5 S     0    10     2  0 -40   - -     0 smpboo ?        00:00:06 watchdog/0
```

```
5 S     0     11      2  0 -40   - -     0 smpboo ?        00:00:05 watchdog/1
```

(d) PID trace on grep:

```
init (1)
lightdm(1094)
lightdm(1800)
 init (1816)
gnome-session(1964)
guake(2247)
bash(13153)
bash(26809)
+-bash(4548)
    |-bash(28430)
    |-bash(29772)
    |-grep(27762)
    `-grep(27837)
```

2. **Process creaation with fork and wait *(Run on my computer)*:**

```
from C0: own PID=21377, parent's PID=21376
from C1: own PID=21378, parent's PID=21376
from P0: own PID=21376, PID of C0=21377, PID of C1=21378, total elapsed time in milliseconds= 2.6610
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <signal.h>
#include <sys/time.h>
#include <sys/wait.h>
#include "fib.c"

int main(int argc, char **argv)
{
        struct timeval start, end;
        long delta;
        pid_t C0, C1, P0;
        P0 = getpid();
        C0 = fork();
        gettimeofday(&start, NULL);
        if (C0 != 0) //parent process
        {
                wait(NULL);
                C1 = fork();
                if (C1 == 0)
                {
                        C1 = getpid();
                        printf("from C1: own PID=%d, parent's PID=%d\n", C1, P0);
                        fib(20);
                        exit(1);
                }
                else
                        wait(NULL);

        }
        else
        {
                printf("from C0: own PID=%d, parent's PID=%d\n", getpid(), P0);
```

```
37                fib(20);
38                exit(1);
39        }
40
41        gettimeofday(&end,NULL);
42        delta = end.tv_usec - start.tv_usec;
43        printf("from P0: own PID=%d, PID of C0=%d, PID of C1=%d, total elapsed time in milliseconds=
              %3.4f\n", P0, C0, C1, delta*.001);
44
45
46        return 0;
47  }
```

3. **Program execution with execl *(Run on my computer)*:**

```
1  Tue Sep 16 11:55:38 MDT 2014
2  rapture :0          2014-09-08 16:53 (:0)
3  rapture pts/22      2014-09-14 11:19 (:0)
4  rapture pts/12      2014-09-14 11:32 (:0)
```

```
1
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <string.h>
5   #include <unistd.h>
6   #include <time.h>
7   #include <signal.h>
8   #include <sys/time.h>
9   #include <sys/wait.h>
10  #include "fib.c"
11
12  int main(int argc, char **argv)
13  {
14        pid_t f = fork();
15
16        if (f != 0) //parent
17        {
18              wait(NULL);
19
20              pid_t f2 = fork();
21              if( f2 != 0)
22                    wait(NULL);
23              else
24              {
25                    fib(20);
26                    execl("/usr/bin/who","who",NULL);
27                    exit(1);
28              }
29        }
30        else
31        {
32              fib(20);
33              execl("/bin/date","date",NULL);
34              exit(1);
35        }
36
37
38        return 0;
39  }
```

4. **Process performance measurement with gettimeofday and getrusage.**

   (a) ***CPU CORES : 4***
   CLK SPEED : 2927.000 MHz
   LOAD AVG : 4.35

   (b) Fill in the blanks

Table 1: mm437_seq.c

N	Elapsed Time	USR Time	SYS Time	USR+SYS Time	# of Context Switches
750	3345	3344	0	3344	4
1500	30320	30315	0	30315	33
3000	297182	297125	0	297125	322

5. Multiple Processes Creation with shared memory

   (a) M = 1

Table 2: mm437_seq.c

N	Elapsed Time	USR Time	SYS Time	USR+SYS Time	# of Context Switches
750	8340	8335	3	8338	11
1500	79480	79431	31	79462	89
3000	706143	705878	127	706005	777

   (b) M = 2

Table 3: mm437_seq.c

N	Elapsed Time	USR Time	SYS Time	USR+SYS Time	# of Context Switches
750	8803	8794	7	8801	11
1500	97662	97594	32	97626	223
3000	712610	712215	245	712460	977

(c) M = 4

Table 4: mm437_seq.c

N	Elapsed Time	USR Time	SYS Time	USR+SYS Time	# of Context Switches
750	12986	12982	1	12983	15
1500	88525	83034	27	83061	629
3000	687655	687600	45	687645	989