1. CPU machine has 4 core.
   Memory Total = 8075196 KB
   Memory Free = 5646208 KB
   Fraction Free = 0.699
   Context Switches since bootup = 8355847
   Processes Forked = 7521

   memeory details are in /proc/meminfo
   cpu core, context switches and process forked are in /proc/stat

2. cpu :
   bottleneck : cpu usage
   reasoning : We used the top command and the cpu usage of this process touched 100%
   justification :There are no system calls, interuupts or traps and thus the only resiurce it requires is the cpu which makes cpu the bottleneck

   cpu-print:
   bottleneck : prinitng on gnome-terminal
   reasoning : while monitoring the processes in the top coomand , the cou usage of gnome-terminal went drastically high when cpu-process was running
   justification : The cpu-print process prints the seconds elapsed on the terminal thus the bottleneck here is the time required by the gnome-terminal to display the output on the screen

   disk:
   bottleneck : memory utilization (sda)
   reasoning : ran the command “iostat -x” and it showed a very high (nearly 100%) memory utilization
   justification : this process is reading from a different file in each of its iteration thus the file has to be obtained from memory each time (cache cant store these many files), thus making the memory utilization a bottleneck

   disk1 :
   bottleneck : cpu usage
   reasoning : We used the top command and the cpu usage of this process touched 100%
   justification : this process is always reading from a same file and thus this file can be accessed from the cache so no time is wasted on obtaining the file again and again from the memory/disk so here agin cpu usae is the highest and thus cpu is bottleneck

3. cpu.c
   amount of time in user mode = 7494
   amount of time in kernel mode = 31

   cpu-print.c
   amount of time in user mode = 265
   amount of time in kernel mode = 1380

   cpu.c spens more time in user mode but cpu-print.c spend more time in kernel mode

   cpu-print.c makes system calls to print on terminal thus more time in kernel mode, as printf will make a system call to write on terminal

4. The program cpu has a lot more nonvoluntary context switches whereas majority of the context switches of disc are voluntary.

The reason for this is that disc is reading from a file in each of its loop which is a blocking system call and thus it volunirly context switches whereas their are no system call or traps or anything in cpu and thus the kernel has to forcibly context switch from it to run other processes thus cpu has higher number of involunatry context switches

5. PID of Bash = 5150
(command : echo $BASHPID)

process tree from pid 1 to bash:

systemd (1) -> lightdm (733) -> lightdm (806) -> upstart (1233) -> gnome-terminal (2514) -> bash (5150)

To obtain this tree we used the pstree -np command which sorts the process tree according to thier pid and then traced the tree to obtainn the process tree from systemd/init (1) to bash (5150)

6. ls and ps are system executables which are just executed bybash shell
whereas cd and history are implemented in bash code itself
we can find this out viathe following command:
type -a [commandname]

7. process id of the new process started :3648

io file descriptor 0 pointing to : /dev/pts/8
io file descriptor 1 pointing to : /tmp/tmp.txt
io file descriptor 0 pointing to : /dev/pts/8

for io redirecting bash just changes the pointer of the these 0/1/2/ file descriptor
by default io descriptor 0 is stdin desciptor 1 is stdout and 2 is stderror. (All are by default set to dev/pts/8 which is the pointer to the terminal)
To redirect io bash changes these pointer from the addrees of the terminal to the address of the required file

8. process id of cpu-print :3946
process id of grep:3947

Here the file descriptors of cpu-print are as follows:
- 0 and 2 are pointing to terminal only
- 1 (which is output) is pointing to pipe:[220749}

and the file descriptors of grep are as follows:
- 1 and 2 are poitinfg to terminal only
- 0 is pointing to pipe:[220749]

thus we can say that pipe works by creating a kind of temporary memory location where the output of the 1st process is stored and from where the second process takes its input