OS LAB 1

Karan Vaidya - 130050019
Sivaprasad S - 130050085

1.
a) Cores : 4 (Can find the information in /proc/cpuinfo)
b) MemTotal : 8146916 kB
MemFree : 5950748 kB (Can find the information in /proc/meminfo)
c) Context Swithces : (ctxt) 4837013 (Can find the information in /proc/stat)
d) Processes forked : 12615 (Same as total number of processes till now, as every process is forked from some other process, Can be found from /proc/stat). vmstat -f also gives the number of forks till now

2.
a) cpu1.c

The bottleneck resource is CPU

commands used - top
top shows nearly 100% CPU usage.
Number of voluntary switches in /proc/id/sched is zero.
iostat, /proc/id/io, iotop shows no I/O operations

Code explanation : There is no syscall in the code. Also there is no read/write from/ to disk. Only performance improvement is possible with improvement in CPU performance.

b) cpu1print.c

The bottleneck resource is I/O device (rendering on screen)

commands used - top, proc/[terminal_pid]/io
top shows very less CPU usage, CPU is not used completely. The disk utilisation from iostat is near zero. The CPU usage of terminal process grew higher(than earlier, not 100%) on running cpu1print.c. The rate of change of io in terminal process has become very high. Hence the bottleneck resource.

The code has many stdout, which requires the screen output (console output), that needs the stream to be parsed for terminal control codes, terminal state to be maintained and render the bitmap. Thus the bottleneck.

c) cpu2.c

Bottleneck resouce : CPU

Commands used - top
top gives nearly 100% CPU.
iostat -dx 1 gives less utilisation of IO to disk.

cpu2 has gettimeofday() which is a virtual system call, so the code is copied to the user space and then it doesn't go to kernel mode or have any voluntary interrupts. Hence CPU is being utilised to the maximum and the bottleneck.

d) disk.c

Bottleneck resource : disk access(I/O)

Commands used - top, iostat -dx 1
top gives very less CPU. The % utilisation of IO to disk is very high (over 99%) as from iostat.
The reads per second is also high. When disk(this program) is not being executed the average time for read requests issued to the device to be served (r_await) from iostat is 0.00 most of the time. but when disk is being run it rises to non zero (around 5ms)

The code has many syscalls to access files from the disk. More over it accesses different files in every iteration and the data is around 20GB, so cannot be fit in main memory.

e) disk1.c

Bottleneck Resource : CPU

Commands used - top
top gives nearly 100% CPU.
IO utilisation is almost zero.

The code only reads from one file (2GB), so all the data can fit in main memory, so not many disk reads. Hence Disk isn't the bottleneck. There are no other blocking syscalls involved and CPU is the bottleneck.

3.
program - usermode, kernel mode (The values correspond to after running the code for different time individually, found from /proc/pid/status)
a)cpu1 - 13965, 0
b)cpu1print- 51,308
c)cpu2 - 2838,0
cpu1 has no syscalls or voluntary interrupts. Hence has very little or no time in kernel mode.
cpu1print has many syscalls. Thus has to spend more time in kernel mode.
cpu2 has gettimeofday() which is a virtual system call, so the code is copied to the user space and then it doesn't go to kernel mode and hence more time spent in user mode.

4.
(found from /proc/pid/sched)
a) cpu1
Voluntary switches : 0
Involuntary switches : 2909

b) disk
Voluntary switches : 6138
Involuntary switches : 419

disk has mostly voluntary context switches as it has to read from different files in the disk everytime and has to use syscalls.
cpu1 has mainly voluntary switches and no involuntary switches as it has no syscalls

Sources :

manual pages for proc, iostat, vmstat
http://stackoverflow.com/questions/17091201/what-is-the-bottleneck-with-standard-streams-displayed-in-terminals