Eric Gonzalez
Troy Edwards

Machine Problem 3:

Part 1:

For part one, we developed the client.cpp program to run with dataserver. cpp by running fork() and a system call to allow dataserver to run from the child process. For measuring the time to send a request and get a response, we used the “chrono” library. We took the system time before and after the request, subtracting the former from the latter to obtain the total time.

Our results were gathered after collecting runtime information from 10,000 requests to both the server and local requests. The topmost table depicts the data collected when handling requests with local functions. This proved to be the faster option and took a fraction of the time needed to handle requests with the server. The time difference is due to the time the request spends between the client and the dataserver.

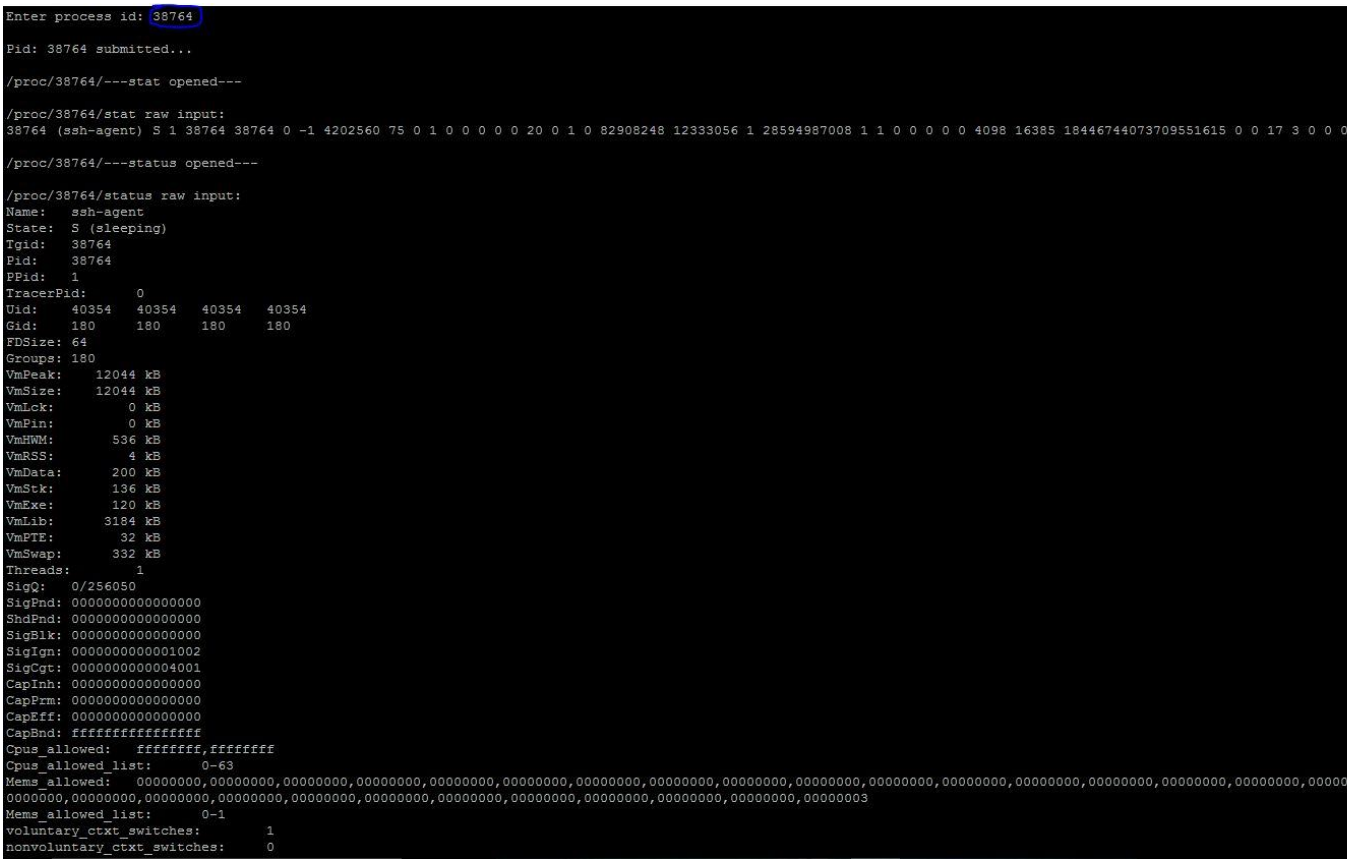
The bottommost table lists data collected on the requests sent to the dataserver. The total runtime was 507.104 milliseconds. For 10,000 requests, the average request time was found to be 50.7104 microseconds. The standard deviation was 1186.1microseconds.

```
*********************************************
               Local Results
*********************************************
runtime           |               18.08 ms
# of requests     |                  10000
avg. request time |                  1.808
standard dev.     |               0.424917
*********************************************
*********************************************
               DataServer Results
*********************************************
runtime           |             507.104 ms
# of requests     |                  10000
avg. request time |                50.7104
standard dev.     |                1186.1
*********************************************
```

Part 2:

Eric Gonzalez
Troy Edwards

1)



Eric Gonzalez
Troy Edwards

```
---Raw input parsed---

*****REQUESTED DATA*****

|CATEGORY 1| Identifiers
--Process ID (PID): 38764
--Parent Process ID (PPID): 1
--Effective User ID (EUID): 40354
--Effective Group ID (EGID): 180
--Real User ID (RUID): 40354
--Real Group ID (RGID): 180
--User File System ID (FSUID): 40354
--Group File System ID (FSGID): 180

|CATEGORY 2| State
--State (STATE): S (sleeping)

|CATEGORY 3| Thread Information
--Thread_Info (NUM_THREADS): 1

|CATEGORY 4| Priorty
--Priority Number (PRIORITY): 20
--Nice Value (NICE): 0

|CATEGORY 5| Time Information
--Time in kernel mode (STIME): 0
--Time in user mode (UTIME): 0
--Time waiting on children in kernel mode (CSTIME): 0
--Time waiting on children in user mode (CUTIME): 0

|CATEGORY 6| Address Space
--Startcode (STARTCODE): 1
--Endcode (ENDCODE): 1
--ESP (KSTKESP): 0
--EIP (KSTKEIP): 0

|CATEGORY 7| Resources
--Number of file descriptors (FDSize): 64
--Number of voluntary context switches (voluntary_ctxt_switches): 1
--Number of nonvoluntary context switches (nonvoluntary_ctxt_switches): 0

|CATEGORY 8| Processors
--Allowed processors (Cpus_allowed): ffffffff,ffffffff
--Last executed processor (PROCESSOR): 3


--Accessing memory map--

--Memory map accessed--

--Memory map: empty--
```

Eric Gonzalez
Troy Edwards

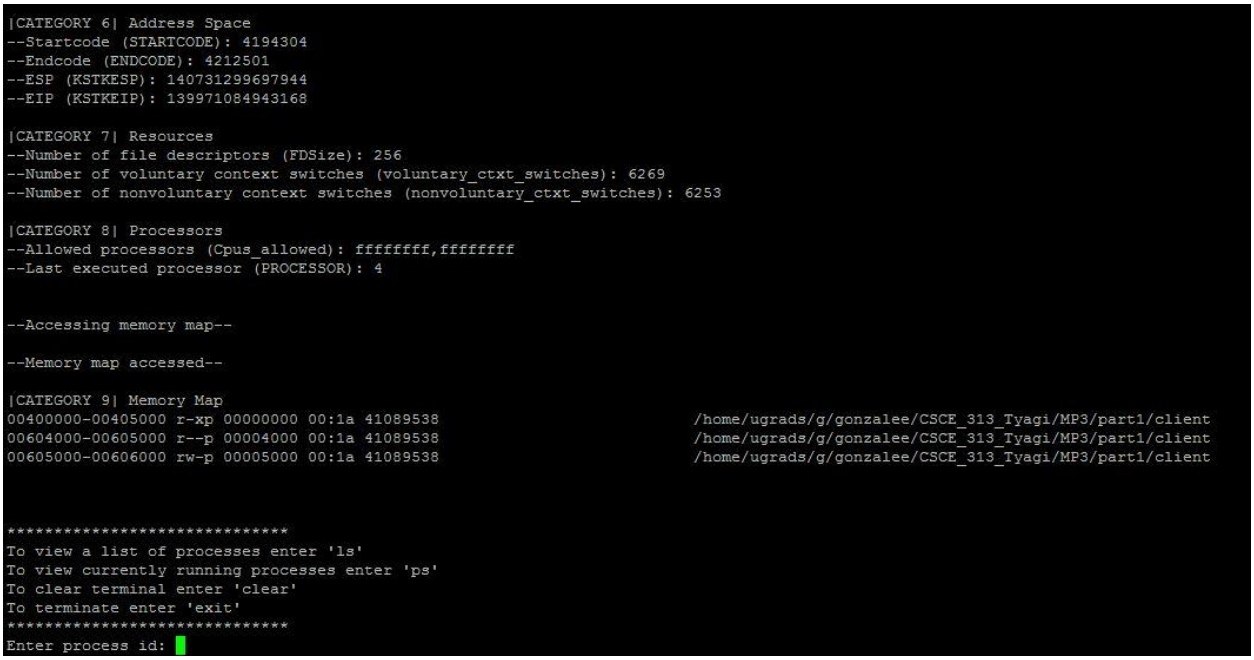
2)

```
******************************
Enter process id: ps

  PID TTY          TIME CMD
24160 pts/0    00:00:00 bash
27727 pts/0    00:00:00 client
27728 pts/0    00:00:00 client
27729 pts/0    00:00:00 dataserver
27741 pts/0    00:00:00 a.out
27742 pts/0    00:00:00 ps

******************************
To view a list of processes enter 'ls'
To view currently running processes enter 'ps'
To clear terminal enter 'clear'
To terminate enter 'exit'
******************************
Enter process id: 27727

Pid: 27727 submitted...

/proc/27727/---stat opened---

/proc/27727/stat raw input:
27727 (client) T 24160 27727 24160 34816 27741 4202496 431 0 0 0 2 5 0 0 20 0 1
071579334661 0 0 17 4 0 0 0 0 0

/proc/27727/---status opened---

/proc/27727/status raw input:
Name:   client
State:  T (stopped)
Tgid:   27727
Pid:    27727
PPid:   24160
TracerPid:      0
Uid:    38938   38938   38938   38938
Gid:    180     180     180     180
FDSize: 256
Groups: 180
VmPeak:    11936 kB
VmSize:    11832 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:      1208 kB
VmRSS:      1208 kB
VmData:      268 kB
VmStk:       172 kB
VmExe:        20 kB
VmLib:      2980 kB
VmPTE:        44 kB
VmSwap:        0 kB
```

Eric Gonzalez
Troy Edwards

```
Threads:        1
SigQ:   1/256050
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 0000000000000000
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: ffffffffffffffff
Cpus_allowed:   ffffffff,ffffffff
Cpus_allowed_list:      0-63
Mems_allowed:   00000000,00000000,00000000,00000000,00000000,00000000,0
0000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,
Mems_allowed_list:      0-1
voluntary_ctxt_switches:        6269
nonvoluntary_ctxt_switches:     6253


---Raw input parsed---

*****REQUESTED DATA*****

|CATEGORY 1| Identifiers
--Process ID (PID): 27727
--Parent Process ID (PPID): 24160
--Effective User ID (EUID): 38938
--Effective Group ID (EGID): 180
--Real User ID (RUID): 38938
--Real Group ID (RGID): 180
--User File System ID (FSUID): 38938
--Group File System ID (FSGID): 180

|CATEGORY 2| State
--State (STATE): T (stopped)

|CATEGORY 3| Thread Information
--Thread_Info (NUM_THREADS): 1

|CATEGORY 4| Priorty
--Priority Number (PRIORITY): 20
--Nice Value (NICE): 0

|CATEGORY 5| Time Information
--Time in kernel mode (STIME): 5
--Time in user mode (UTIME): 2
--Time waiting on children in kernel mode (CSTIME): 0
--Time waiting on children in user mode (CUTIME): 0

|CATEGORY 6| Address Space
--Startcode (STARTCODE): 4194304
```

Eric Gonzalez
Troy Edwards



3) The kernel usually only checks the effective user ID. When a process tries to open a file, the effective user ID is checked by the kernel which then decides to grant access to the file or not. The real user ID matters when one wishes to change the effective user ID of a running process. A situation where the real user ID and effective ID are different is when a process is running as root and a particular user sends a rquest to access a file. In this situation, the effective user ID will change from root to the user and check to see if they have access to the desired file. After all operations have been carried out, the effective ID returns to its original value using the real user ID.
4) Most files in /proc are read only because changing them would alter information that the kernel relies on, potentially causing problems.
5) The task_struct is allocated by the slab allocator to provide object reuse and cache coloring. Most kernel code dealing with processes works directly with task_struct, so it's crucial for the computer to quickly look up the task_struct of the process that is currently running.