COMP3000 - Exercise 3 (Individual)
Process Manager - Scheduler - Scheduling

Winter 2018

Modify MINIX scheduling such that it implements fair-share scheduling. With fair-share scheduling, the CPU is shared by a set of processes according to the group to which they belong. For example, if there are three groups G1, G2 and G3. Each group should get one third of the CPU time with fair-share scheduling. Let us suppose that there is one process in G1, three processes in G2, and six processes in G3. The process in group G1 should get 33% of the CPU time, each process in G2 should get 11% of the CPU time and each process in G3 should get 5.5% of the CPU time.

Modify the scheduler such that all processes use fair-share scheduling. Group membership is determined by the process group identification (field `mp_procgrp` of `struct mproc`), file `/usr/src/minix/servers/pm/mproc.h`.
Use the following two test programs:

```
/* File: tester1.c
*
* A CPU-bound process (no children).
*/
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
   int i, j;
   i = 1;
   while (1) {
      i = j; j = i;
   }
}

/* File: tester2.c
*
* A CPU-bound process with one child.
*/
#include <stdio.h>
```

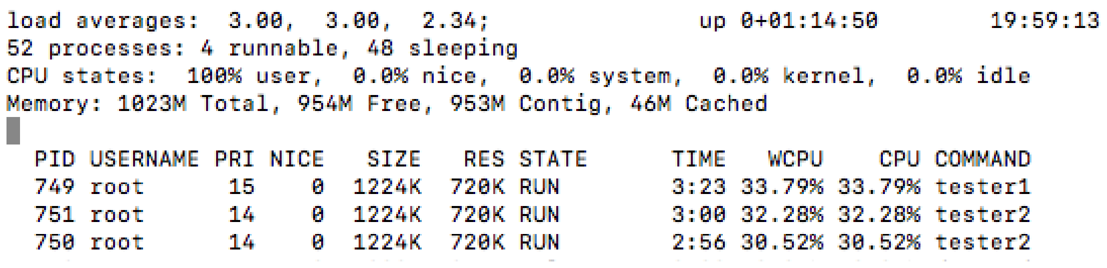
```
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
   int i, j;
   fork();
   i = 1;
   while (1) {
      i = j; j = i;
   }
}
```

Start each program in different terminals using `./tester1 &` and `./tester2 &`. The command `ps` can be used to confirm that processes are created in two different groups:

```
minix# ps -o pid,ppid,pgid,command
PID PPID PGID COMMAND
667  666  667 -sh
750  667  667 ./tester2
751  750  667 ./tester2
678  677  678 -sh
749  678  678 ./tester1
```

The field `pgid` stands for process group ID.

In original MINIX, the command `top` shows that each of the CPU-bound process gets approximately one third of the CPU time.

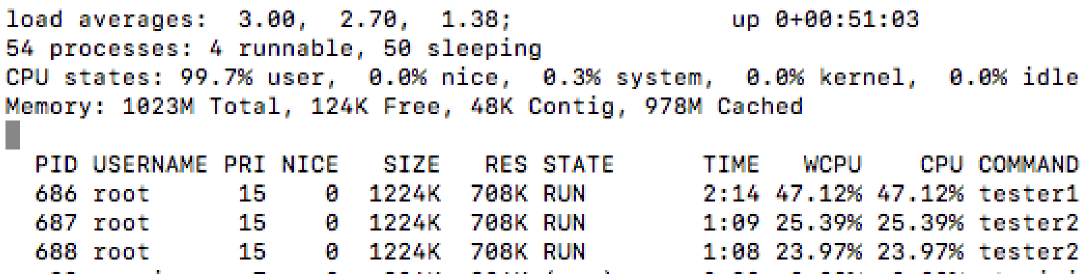


```
load averages:  3.00,  3.00,  2.34;              up 0+01:14:50        19:59:13
52 processes: 4 runnable, 48 sleeping
CPU states:  100% user,  0.0% nice,  0.0% system,  0.0% kernel,  0.0% idle
Memory: 1023M Total, 954M Free, 953M Contig, 46M Cached

  PID USERNAME PRI NICE   SIZE   RES STATE      TIME   WCPU    CPU COMMAND
  749 root      15    0  1224K  720K RUN        3:23 33.79% 33.79% tester1
  751 root      14    0  1224K  720K RUN        3:00 32.28% 32.28% tester2
  750 root      14    0  1224K  720K RUN        2:56 30.52% 30.52% tester2
```

The process associated with `tester1` is one group. When started in a different terminal, the two processes associated with `tester2` is another group. In MINIX together with your implementation of fair-share scheduling, the command `top` should show that each of the group gets approximately one half of the CPU time, while each of the two processes in the second group should get approximately of fourth of the CPU time.

```
minix# ps -o pid,ppid,pgid,command
PID PPID PGID COMMAND
687  631  631 ./tester2
688  687  631 ./tester2
686  682  682 ./tester1
```

There are two groups. Each of them should get close to 50% of the CPU time, while each of the two processes in the second group should get close to 25% of the CPU time.



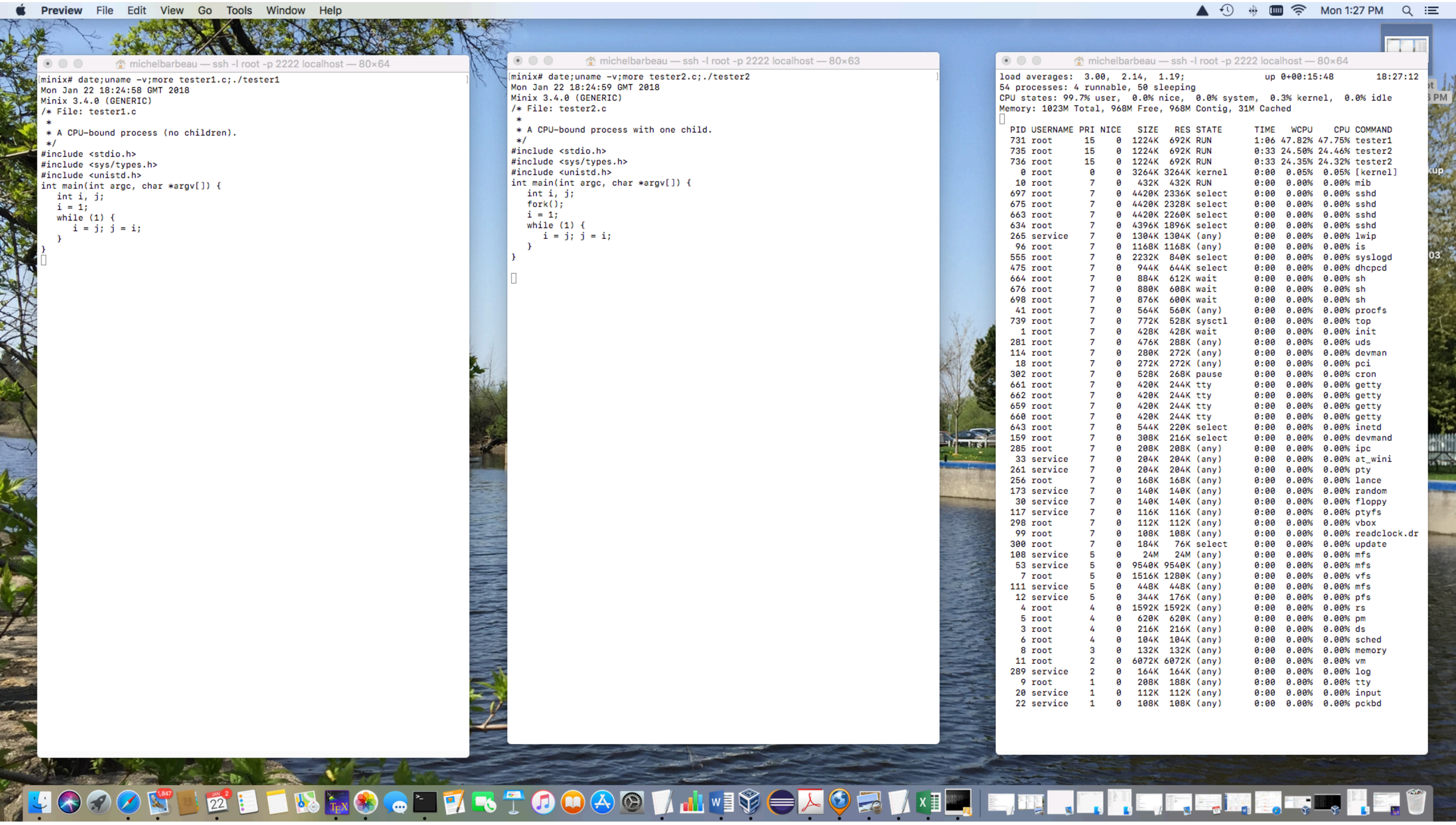
```
load averages:  3.00,  2.70,  1.38;                up 0+00:51:03
54 processes: 4 runnable, 50 sleeping
CPU states: 99.7% user,  0.0% nice,  0.3% system,  0.0% kernel,  0.0% idle
Memory: 1023M Total, 124K Free, 48K Contig, 978M Cached

  PID USERNAME PRI NICE   SIZE   RES STATE      TIME   WCPU    CPU COMMAND
  686 root      15    0  1224K  708K RUN        2:14 47.12% 47.12% tester1
  687 root      15    0  1224K  708K RUN        1:09 25.39% 25.39% tester2
  688 root      15    0  1224K  708K RUN        1:08 23.97% 23.97% tester2
```

Due date: February 4. This exercise must be done in the C programming language under MINIX 3.4. Submit your work on cuLearn. Submit a single *tar.gz* file. Include a README.txt file containing a report about your work (describe every change you made to the system code and where you made it). Your are responsible for the completeness of your submission. Your are responsible for submitting your work on time. Your submission must include a screenshot showing evidence that you code is working (see the attached example). Open three terminals, call the following sequence of commands:

1. `date;uname -v;more tester1.c;./tester1`

2. `date;uname -v;more tester2.c;./tester2`

3. `top`


Preview File Edit View Go Tools Window Help
Mon 1:27 PM
michelbarbeau — ssh -l root -p 2222 localhost — 80×64
michelbarbeau — ssh -l root -p 2222 localhost — 80×63
michelbarbeau — ssh -l root -p 2222 localhost — 80×64

```
minix# date;uname -v;more tester1.c;./tester1
Mon Jan 22 18:24:58 GMT 2018
Minix 3.4.0 (GENERIC)
/* File: tester1.c
 *
 * A CPU-bound process (no children).
 */
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
   int i, j;
   i = 1;
   while (1) {
      i = j; j = i;
   }
}
```

```
minix# date;uname -v;more tester2.c;./tester2
Mon Jan 22 18:24:59 GMT 2018
Minix 3.4.0 (GENERIC)
/* File: tester2.c
 *
 * A CPU-bound process with one child.
 */
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
   int i, j;
   fork();
   i = 1;
   while (1) {
      i = j; j = i;
   }
}
```

```
load averages:  3.00,  2.14,  1.19;                up 0+00:15:48        18:27:12
54 processes: 4 runnable, 50 sleeping
CPU states: 99.7% user,  0.0% nice,  0.0% system,  0.3% kernel,  0.0% idle
Memory: 1023M Total, 968M Free, 968M Contig, 31M Cached

  PID USERNAME PRI NICE   SIZE   RES STATE      TIME   WCPU    CPU COMMAND
  731 root      15    0  1224K  692K RUN        1:06 47.82% 47.75% tester1
  735 root      15    0  1224K  692K RUN        0:33 24.50% 24.46% tester2
  736 root      15    0  1224K  692K RUN        0:33 24.35% 24.32% tester2
    0 root       0    0  3264K 3264K kernel     0:00  0.05%  0.05% [kernel]
   10 root       7    0   432K  432K RUN        0:00  0.00%  0.00% mib
  697 root       7    0  4420K 2336K select     0:00  0.00%  0.00% sshd
  675 root       7    0  4420K 2328K select     0:00  0.00%  0.00% sshd
  663 root       7    0  4420K 2260K select     0:00  0.00%  0.00% sshd
  634 root       7    0  4396K 1896K select     0:00  0.00%  0.00% sshd
  265 service    7    0  1304K 1304K (any)      0:00  0.00%  0.00% lwip
   96 root       7    0  1168K 1168K (any)      0:00  0.00%  0.00% is
  555 root       7    0  2232K  840K select     0:00  0.00%  0.00% syslogd
  475 root       7    0   944K  644K select     0:00  0.00%  0.00% dhcpcd
  664 root       7    0   884K  612K wait       0:00  0.00%  0.00% sh
  676 root       7    0   880K  608K wait       0:00  0.00%  0.00% sh
  698 root       7    0   876K  600K wait       0:00  0.00%  0.00% sh
   41 root       7    0   564K  560K (any)      0:00  0.00%  0.00% procfs
  739 root       7    0   772K  528K sysctl     0:00  0.00%  0.00% top
    1 root       7    0   428K  428K wait       0:00  0.00%  0.00% init
  281 root       7    0   476K  288K (any)      0:00  0.00%  0.00% uds
  114 root       7    0   280K  272K (any)      0:00  0.00%  0.00% devman
   18 root       7    0   272K  272K (any)      0:00  0.00%  0.00% pci
  302 root       7    0   528K  268K pause      0:00  0.00%  0.00% cron
  661 root       7    0   420K  244K tty        0:00  0.00%  0.00% getty
  662 root       7    0   420K  244K tty        0:00  0.00%  0.00% getty
  659 root       7    0   420K  244K tty        0:00  0.00%  0.00% getty
  660 root       7    0   420K  244K tty        0:00  0.00%  0.00% getty
  643 root       7    0   544K  220K select     0:00  0.00%  0.00% inetd
  159 root       7    0   308K  216K select     0:00  0.00%  0.00% devmand
  285 root       7    0   208K  208K (any)      0:00  0.00%  0.00% ipc
   33 service    7    0   204K  204K (any)      0:00  0.00%  0.00% at_wini
  261 service    7    0   204K  204K (any)      0:00  0.00%  0.00% pty
  256 root       7    0   168K  168K (any)      0:00  0.00%  0.00% lance
  173 service    7    0   140K  140K (any)      0:00  0.00%  0.00% random
   30 service    7    0   140K  140K (any)      0:00  0.00%  0.00% floppy
  117 service    7    0   116K  116K (any)      0:00  0.00%  0.00% ptyfs
  298 root       7    0   112K  112K (any)      0:00  0.00%  0.00% vbox
   99 root       7    0   108K  108K (any)      0:00  0.00%  0.00% readclock.dr
  300 root       7    0   184K   76K select     0:00  0.00%  0.00% update
  108 service    5    0    24M   24M (any)      0:00  0.00%  0.00% mfs
   53 service    5    0  9540K 9540K (any)      0:00  0.00%  0.00% mfs
    7 root       5    0  1516K 1280K (any)      0:00  0.00%  0.00% vfs
  111 service    5    0   448K  448K (any)      0:00  0.00%  0.00% mfs
   12 service    5    0   344K  176K (any)      0:00  0.00%  0.00% pfs
    4 root       4    0  1592K 1592K (any)      0:00  0.00%  0.00% rs
    5 root       4    0   620K  620K (any)      0:00  0.00%  0.00% pm
    3 root       4    0   216K  216K (any)      0:00  0.00%  0.00% ds
    6 root       4    0   104K  104K (any)      0:00  0.00%  0.00% sched
    8 root       3    0   132K  132K (any)      0:00  0.00%  0.00% memory
   11 root       2    0  6072K 6072K (any)      0:00  0.00%  0.00% vm
  289 service    2    0   164K  164K (any)      0:00  0.00%  0.00% log
    9 root       1    0   208K  188K (any)      0:00  0.00%  0.00% tty
   20 service    1    0   112K  112K (any)      0:00  0.00%  0.00% input
   22 service    1    0   108K  108K (any)      0:00  0.00%  0.00% pckbd
```