```
/*
Class: CPSC 346-01 & CPSC 346-02
Team Member 1: Nicholas Walker (Section 1)
Team Member 2: Brett Barinaga (Section 2)
GU Username of project lead: nwalker
Pgm Name: proj3.c
Pgm Desc: exploraiton of the proc file system
Usage: 1) standard:  ./a.out -s
Usage: 2) history:  ./a.out -h
Usage: 3) load:  ./a.out -l
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void standard();
void history();
void load();

int main(int argc, char* argv[])
{
 if (argc != 2)
  {
   fprintf(stderr, "Error: Options required\n");
   fprintf(stderr, "usage: ./a.out -s|-h|-l\n\n");
   exit(EXIT_FAILURE);
  }

 if (!strcmp(argv[1],"-s"))
  standard();
 else if (!strcmp(argv[1],"-h"))
  history();
 else if (!strcmp(argv[1],"-l"))
  load();
else
 {
  fprintf(stderr, "Incorrect command line argument. Use -s, -h or -
l\n");
  exit(EXIT_FAILURE);
 }
}
/*
pre: none
post: displays CPU vendor_id, model name, and OS version
*/
void standard()
{
 char ch;
 FILE* ifp;
```

```
 char str[80];

 /*
 I've deliberately used two different mechanisms for writing to the
console.
 Use whichever suits you.
 strstr locates a substring
 */

 ifp = fopen("/proc/cpuinfo","r");
 while (fgets(str,80,ifp) != NULL)
  if (strstr(str,"vendor_id") || strstr(str,"model name"))
   puts(str);
 fclose (ifp);

 ifp = fopen("/proc/version","r");
 while ((ch = getc(ifp)) != EOF)
  putchar(ch);
 fclose (ifp);


}

/*
pre: none
post: displays time since the last reboot (DD:HH:MM:SS), time when the
system was last
      booted, number of processes that have been creates since the
last reboot
      Hint: strftime could be useful
*/
void history()
{
  char ch;
  FILE* ifp;
  char str[80];
  char uptime[80];
  int i = 0;
  char rtc[80];
  char hour[2];
  char minute[2];
  char second[2];
  int secondsnow = 0;
  ifp = fopen("/proc/driver/rtc","r");
  while (fgets(str,80,ifp) != NULL)
  if (strstr(str,"rtc_time"))
    strncpy(rtc, str, 19);
  fclose (ifp);

  hour[0] = rtc[11];
```

```
  hour[1] = rtc[12];
  minute[0] = rtc[14];
  minute[1] = rtc[15];
  second[0] = rtc[17];
  second[1] = rtc[18];

  int hourint;
  hourint = (hour[0]-'0')*10 + (hour[1]-'0');
  secondsnow += hourint*60*60;

  int minuteint;
  minuteint = (minute[0]-'0')*10 + (minute[1]-'0');
  secondsnow += minuteint*60;

  int secondint;
  secondint = (second[0]-'0')*10 + (second[1]-'0');
  secondsnow += secondint;

//////////
  ifp = fopen("/proc/uptime", "r");
  while ((ch = getc(ifp)) != ' ')
  {
    uptime[i] = ch;
    i++;
  }
  fclose (ifp);
  double timeup;
  sscanf(uptime, "%lf", &timeup);
  int up = timeup;
  int days;
  int hours;
  int minutes;
  int seconds;
  days = timeup/(60*60*24);
  timeup -= days*(60*60*24);
  hours = timeup/(60*60);
  timeup -= hours*(60*60);
  minutes = timeup / 60;
  timeup -= minutes*60;
  seconds = timeup;

  fprintf(stderr, "\nTime since last reboot: ");
  if(days < 10)
    printf("0%d:", days);
  else
    printf("%d:", days);

  if(hours < 10)
    printf("0%d:", hours);
  else
```

```
    printf("%d:", hours);

  if(minutes < 10)
    printf("0%d:", minutes);
  else
    printf("%d:", minutes);

  if(seconds < 10)
    printf("0%d\n", seconds);
  else
    printf("%d\n", seconds);

  int timesince;
  timesince = secondsnow - up;

  int newhour;
  int newmin;
  int newsec;

  newhour = timesince / (60 * 60);
  timesince -= (newhour * (60 * 60));
  newmin = timesince / 60;
  timesince -= (newmin * 60);
  newsec = timesince;

  fprintf(stderr, "Time when the system was last booted (UTC): ");
  printf("%d:", newhour);
  printf("%d:", newmin);
  printf("%d\n", newsec);



  ifp = fopen("/proc/stat", "r");
  while (fgets(str,80,ifp) != NULL)
    if (strstr(str,"processes"))
      puts(str);
  fclose (ifp);
}

/*
pre: none
post: displays total memory, available memory, load average (avg.
number of processes over the last minute)
*/
void load()
{
  char ch;
  FILE* ifp;
  char str[80];
```

```
  ifp = fopen("/proc/meminfo", "r");
  while (fgets(str,80,ifp) != NULL)
    if (strstr(str,"MemTotal") || strstr(str,"MemAvailable"))
      puts(str);
  fclose (ifp);

  fprintf(stderr, "Load Average: ");

  ifp = fopen("/proc/loadavg", "r");
  while ((ch = getc(ifp)) != ' ')
      putchar(ch);
  fclose (ifp);
}
```