1 Grading

This homework is worth 100 points. Incomplete answers get partial credit commensurate to the completeness of the answer.

Should there be any indication of plagiarism, the grade for the assignment will be 0. Administrative action may be taken.

2 Assignment

1. (10 points) Five jobs (A, B, C, D, and E), arrive in alphabetical order in the following quanta, respectively: 1, 2, 2, 5, 7. They have estimated running times of 5, 4, 4, 3, and 6 minutes. Their priorities are 1, 3, 5, 3, and 1, respectively, with 5 being the lowest priority. For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead and assume a quantum of 1 minute. Show your work for partial credit.

    - Round robin.
    - Priority scheduling.
    - First-come, first-served
    - Shortest-job first.

2. (10 pts) The aging algorithm with $a = 1/2$ is being used to predict run times. The previous four runs, from oldest to most recent, are 8, 12, 9, and 11 msec. What is the sequence of predicted runtimes starting with the first of the observed runs (8), ending with the predicted runtime after the last observed run (11). (See Tanenbaum's section on Scheduling in Real-Time Systems, Ch. 2)

3. (10 pts) A real-time system needs to handle one voice call that runs every 8 msec and consumes 4 msec of CPU time per burst, plus a video at 30 frames/sec, with each frame requiring 20 msec of CPU time. Is this system schedulable? Explain.

4. (10 pts) Consider a swapping system in which memory consists of the following hole sizes in memory order: 2KB, 8KB, 8KB, 6KB, 20KB, 8KB, 24KB, and 32KB. Which hole is taken for successive segment requests of 4KB, 8KB, and 16KB when each of the following algorithms is used:

   - First fit.
   - Best fit.
   - Next fit.
   - Worst fit.

5. (10 pts) For each of the following decimal virtual addresses, compute the virtual page number and offset for a 4KB page and for an 8KB page: 4097, 16390, 66666, 1000000.

6. (10 points) A computer has four page frames. The time of loading, time of last access, and the R and M bits for each page are shown below (with times in clock ticks):

Page	*Loaded*	*Last Ref.*	R	M
0	100	220	1	1
1	90	210	0	1
2	140	200	1	0
3	110	190	0	1

   Which page will NRU replace?
   Which page will FIFO replace?
   Which page will LRU replace?
   Which page will second chance replace?

7. (10 pts) A computer with a 64-bit address uses a three-level page table. Virtual addresses are split into a top-level page table field, a 17-bit second-level page table field, a 17-bit third-level page table field. If we use 16KB pages, how many entries can the top-level page table maximally have? Show your work for full credit.

8. (10 pts) Write a C program that reverses the bytes of a given arbitrary length file and outputs them to another specified file, which should be overwritten if it already exists. So, invoking your program as follows
   `./reverse infile outfile`
   will result in the first byte of `outfile` being the last byte of `infile`, and so on until the last byte of `outfile` is the first byte of `infile`. The given filenames are just an example, make your own `infile`. You must use `fseek()/fread()/fwrite()`.

9. (10 pts) Write another C program that reverses the bytes of a given arbitrary length file in-place, as in, the contents of the file are swapped without a temporary file and without producing a second file. Use `mmap()`. E.g. `./reverse2 infile` results in the contents of `infile` being reversed after the call, and calling `./reverse2 infile` twice should result in the same file as you started with.

10. (10 pts) Compare the performance of `./reverse` with `./reverse2`. Use the timing function below to target your timing to just the portion of the program which actually does the reversing and produces output. Get the averages of a bunch of runs.

```
#include <sys/time.h>
#include <unistd.h>
#include <assert.h>

// you may use this to convert the contents of the timeval struct to ns
long nanosec(struct timeval t){
  return((t.tv_sec*1000000+t.tv_usec)*1000);
}

int main(){
  int res;
  struct timeval t1, t2;

  res=gettimeofday(&t1,NULL); assert(res==0);
  //stuff you want to measure might go here
  res=gettimeofday(&t2,NULL);   assert(res==0);
  //find average time here
}
```

3 What to turn in

Turn in an archive (tar, tgz, or zip) containing the answers to the questions above and the code associated with your answers (make sure your code compiles and runs on os.cs.siue.edu!