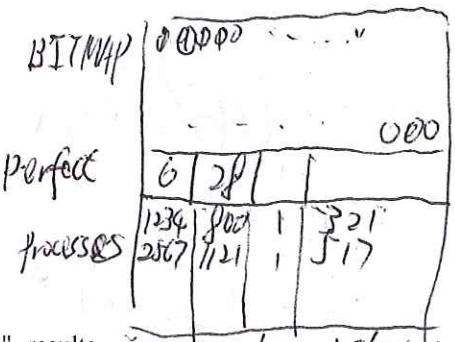
% manage &
% compute 1000 &
% compute 5000 &
% report
% compute 10000 &
% report -k

>1 processes @ see the same blank spot and wants to use it:
in bit map: very frequent, min penalty
in perfect/process: not frequent, huge penalty



solution? compute → task → manage → write perfect / pid → @

@ compute → manage → return pid → compute → find in shared mem.

shared memory could use semaphors to prevent multiple process to test the same number, but is system call, too expensive. so, don't use it for testing

However

Assignment 3 CS 551

Shared Memory and Signals on System V

Due Friday November 11, 2016 at 10PM

For this assignment you will write 3 related programs to "manage", "report", and "compute" results stored in shared memory. You should hand in files manage.c, report.c, and compute.c to implement the functions described below.

Compute's job is to compute perfect numbers. It takes one command line argument, which is the first number to test. It tests all numbers starting at this point, subject to the constraints below. There may be more than one copy of compute running simultaneously.

Manage's job is to maintain the shared memory segment. The shared segment is where the compute processes post their results. It also keeps track of the active "compute" processes, so that it can signal them to terminate.

Report's job is to read the shared memory segment and report on the perfect numbers found, the total number tested, and for each processes currently computing the number tested, skipped, and found. It should also give a total of these three numbers that includes processes no longer running. If invoked with the "-k" switch, it also is used to inform the Manage process to shut down computation.

@ total: test, fnd, skp

@ test, fnd, skip for each process

@ total should also keep track of terminated process

The shared memory segment should contain the following data:

(1) A bit map large enough to contain 2^25 bits. If a bit is off it indicates the corresponding integer has not been tested.

2^22 bytes

(2) An array of integers of length 20 to contain the perfect numbers found.

(3) An array of "process" structures of length 20, to summarize data on the currently active compute processes. This structure should contain the pid, the number of perfect numbers found, the number of candidates tested, and the number of candidates not tested. "Compute" should never test a number already marked in the bitmap.

Compute processes are responsible for updating the bitmap, as well as their own process statistics. However, because of the possible conflicts, "manager" must initialize their process entry for each compute process. You may use your favorite IPC scheme for "compute" registering itself with "manager". Similarly, "compute" must request "manager" update the array of perfect numbers, when it finds one.

exit → not include report since it don't keep running

Processes that hit the end should wrap around, but stop at their starting point. All processes should terminate cleanly on INTR, QUIT, and HANGUP signals. For "compute" processes this means they delete their process entry from the shared memory segment and then terminate. For "manager" it means he sends an INTR signal to all the running computes, sleeps 5 seconds, and then deallocates the shared memory segment, and terminates.

'oldsignals.c

When the -k is flag is used on "report", report sends an INTR to manager to force the same shutdown procedure.

same as manage shut down?

So that there are no conflicts between users, use the last 5 digits of your phone number as a key for the shared memory segment. Also so as not use up unnecessary resources during the debugging phase, use the ps, ipcs, and ipcrm commands to make sure you have not left extraneous processes or shared memory segments allocated.

manage assign one / row in process for "total". so at most 19 process

report -k add the number of processes to total, then kill the processes & (only report accumulated total, since no more active processes (all killed))

How to make sure right (20 rows)

use the manage process's pid for the total line, so it can be killed with that pid.