## CS 354 Fall 2025

## Homework 2 [20 pts]

**Juan Florez**

1. Suppose only three Xinu processes are running as follows:

Suppose that process C calls send() to send the message “54” to process B.

1. What will the states of processes A and B be immediately after process C calls send()?

Process A: READY.

Process B: READY.

1. If process C then calls sleep() to sleep for 1 second, what will the states of processes A and B be immediately after process C starts to sleep? (i.e., just after the context switch as either A or B starts executing.)

Process A: CURRENT.

Process B: READY.

1. Suppose only three Xinu processes are running as follows:

(i.e., as above except the priorities of A and B are reversed)

Now suppose that process C calls send() to send the message “54” to process B.

1. What will the states of processes A and B be immediately after process C calls send()?

Process A: READY.

Process B: READY.

1. If process C then calls sleep() to sleep for 1 second, what will the states of processes A and B be immediately after process C starts to sleep? (i.e., the states before A or B start executing.)

Process A: READY.

Process B: CURRENT.

1. Assume only one Xinu process, X, is running. Assume X has been running for a while, has created and resumed six other processes, and all six of the other processes have exited normally by returning from their top-level function. Now suppose X creates and resumes a process Y that has a lower priority than X. Immediately after resuming Y, process X calls sleep(1). Process Y runs, calls send() with message “101” to process X and then exits. Suppose that after Y exits, process X calls receive().
2. Will the receive() call block process X, or will the call return immediately without blocking? Explain.

The call returns immediately without blocking, because process Y sent (101) to X message queue before exiting.

1. Is the call to receive() in process X guaranteed to return “101”? Explain why or why not.

Yes, it is guaranteed to return (101), because X message queue contains only the message (101) from Y.

INITIAL CONDITIONS USED FOR QUESTIONS 4, 5, and 6:

* Suppose that at the beginning of the main process execution, the free memory consists of a single block that contains 100,000 bytes.
* The lowest address in the block of free memory is location L.
* Assume that no other processes are running.

1. Consider low-level memory allocation in Xinu.
2. Suppose Xinu has started, and no other process has allocated or freed memory. Suppose main calls getmem(5000) which returns an address, addr1, and then calls getmem(5000) again, which returns address addr2. What is the value of addr2?

The value of addr2 = L + 5000

1. Continuing from 4(a), suppose the main process calls getmem(5000) eighteen more times. Will all eighteen calls succeed, or will the last call fail because getmem must reserve some bytes of the free space for the head of the free memory list? Explain.

All calls succeed, because after two calls, 90000 bytes remain and the 18 calls request 90.000 bytes, which fits without needing extra space.

1. Go back to the initial conditions above, and suppose the main makes ten calls of getmem(10000). For purposes of this answer, assume all ten calls succeed and return ten blocks. Now suppose that main makes ten calls to freemem (one for each previously allocated block) to free the first half of the allocated block. How much free memory remains?

50000 bytes

1. If after part (a), main calls getmem(15000), will the call succeed or fail? Explain.

The call succeed, because the free memory is sufficient to allocate 15000 bytes.

1. Go back to the initial conditions above, and suppose that main creates and resumes four low-priority processes in the following order: A, B, C, and D. Each of the four processes has a stack of size 24,000 bytes. Suppose main sleeps for one second. Process B starts executing and exits. The other three continue to use the CPU without any exiting. Now suppose main wakes up, resumes its execution and calls getmem(8000). If the call succeeds, what is the address that getmem returns? And if the call fails, explain why.

L + 24000, the four processes allocate 96000 bytes, leaving 4000 bytes at L + 96000, also proccees B exit frees 24000 bytes at L + 24000, and getmem(8000) allocates from this block using first-fit.

1. Suppose main makes another call to getmem(2000) after part (a). If the call succeeds, what is the address that getmem returns? And if the call fails, explain why.

L + 96000, after part a, free blocks are 4000 bytes at L + 96000 and 16000 bytes at L + 32000, finally getmem(2000) allocates from L + 96000 using first-fit.

1. Consider a Unix system running on unusual hardware. Assume that the system uses paging with a page size of 2048 bytes. Assume each process is allocated a 24-bit address space.
2. How many frames will be needed in memory to hold the entire address space for a process?

8192 frames.

1. If a page table entry contains 8 bytes, how many bytes of memory will be needed to hold the page tables for NPROC processes? Express your answer in bytes and Megabytes (multiples of 1024\*1024 bytes).

65530 x NPROC bytes, 0.0623 x NPROC MB

1. Consider the following declarations:

#pragma pack(2)

struct x {

int32 a;

int16 b;

char c[1024];

int16 d;

int16 e;

};

struct x xa[400];

#pragma pack()

#pragma pack(4)

struct y {

int32 a;

int16 b;

char c[1024];

int16 d;

int16 e;

};

struct y ya[400];

#pragma pack()

1. Assuming a page size of 4096 bytes (4KB page size), how many pages will xa occupy? Explain.

102 pages, because struct x size is 1034 bytes and array xa is 400, 400 x 1034 = 413600 bytes and page = 4096, so 413600 / 4096 = 102.

1. Assuming a page size of 4096 bytes (4KB page size), how many pages will ya occupy? Explain.

102 pages, because struct y size is 1040 bytes and array ya is 400, 400 x 1040 = 416000 bytes and page = 4096, so 416000 / 4096 = 102.

1. Consider a traditional Unix system that uses paging (i.e., the address space is not randomized for security). Suppose a process running on such a system is only given six frames in memory. At one point you stop the process and examine the resident pages.
2. You find that one of the six frames is filled with a page from an address below \*etext\*. Does that indicate a problem, or do you expect such a page? Explain.

Expected, because pages below \*etext\* are form the text segment, which must be in memory for execution.

1. You also find a page with a very high address, and the remaining frames are filled with pages with addresses between \*etext\* and \*end\*. Do the remaining resident pages make sense, or has an error occurred? Explain.

Makes sense, because High address page is from the stack, and pages between \*etext\* and \*end\* are from the data segment.

1. A user purchases a computer; to save money, he chooses a very inexpensive and very small memory. The operating system uses paging. The user runs a scientific application written in FORTRAN, which stores arrays in column-major order. The application stores double-precision floating point values in an N x N array, where N is very large, and performs its calculation by adding and multiplying successive values across each row of the array, one row at a time. Performance is terrible.
2. What classic paging problem will the application cause? Explain.

Row wise access of a column major array (FORTRAN) causes poor locality, page faults in small memory. That is thrashing.

1. If the application is rewritten in C, will it run faster? Why or why not?

It may improve but not guaranteed faster, because C row major storage with row wise access, improving locality but small memory still.

## Submission

You must submit a directory named hw2that contains a PDF file (created using dedicated PDF creator or converter):

* questions.pdf, containing your answers.

Go to the directory where hw2 is a subdirectory.

For example, if /homes/alice/cs354/hw2 is your directory structure, go to /homes/alice/cs354.

Type the following command to submit the directory with turnin:

turnin -c cs354 -p hw2 hw2

Be sure the file inside the directory is named exactly questions.pdf.

You can check/list the submitted files using  
  
turnin -c cs354 -p hw2 -v