- **Match the elements of C program to their place in memory**
  - Global variables-> Data
  - Local Static variables-> Data
  - Global Static variables-> Data
  - Local Variables-> Stack
  - Arguments-> Stack
  - Malloced Memory-> Heap
  - Function code-> Code
  - Code of main()-> Code
  - #include files-> No memory needed
  - #define MACROS-> No Memory needed
  - -> Main_Code

C program to segment (Matching)

- **Match the File descriptors to their meaning**
  - 0-> Standard Input
  - 1-> Standard output
  - 2-> Standard error

FDs to meaning (Matching)

- **Match the MACRO with it's meaning**
  - PHYSTOP-> 224 MB
  - KERNBASE-> 2 GB
  - KERNLINK-> 2.224 GB
  - -> 2.1 GB
  - -> 2 MB

Meaning of MACROS in MM (wrong choice 2.224) (Matching)

- **Match the names of PCB structures with kernel**
  - xv6-> struct proc
  - linux-> struct task_struct
  - -> struct process
  - -> struct task_structure
  - -> struct process_struct

PCB names (Matching)

- **Arrange in correct order, the files involved in execution of system call**
  - usys.S-> 1
  - vectors.S-> 2
  - trapasm.S-> 3
  - trap.c-> 4

Syscall order correctly (Matching)

- **A process blocks itself means**

  a. (100%) The kernel code of system call, called by the process, moves the process to a waiting queue and calls scheduler

  b. (0%) The application code calls the scheduler

  c. (0%) The kernel code of system call calls scheduler

  d. (0%) The kernel code of an interrupt handler, moves the process to a waiting queue and calls scheduler

Blocking means (Multiple choice / One answer only)

- **What will be the output of this program int main() { int fd; printf("%d ", open("/etc/passwd", O_RDONLY)); close(1); fd = printf("%d ", open("/etc/passwd", O_RDONLY)); close(fd); fd = printf("%d ", open("/etc/passwd", O_RDONLY)); }**

  a. (100%) 3 1 1

  b. (0%) 3 4 5

  c. (0%) 3 1 2

  d. (0%) 1 1 1

  e. (0%) 2 2 2

  f. (0%) 3 3 3

FD output (Multiple choice / One answer only)

- **Which of the following is not a task of the code of swtch() function**

  a. (50%) Save the return value of the old context code

  b. (50%) Change the kernel stack location

  c. (0%) Save the old context

  d. (0%) Load the new context

  e. (0%) Jump to next context EIP

  f. (0%) Switch stacks

Not done by swtch() (Multiple choice)

- **Which of the following state transitions are not possible?**

  a. (33.33333%) Ready -> Terminated

  b. (33.33333%) Waiting -> Terminated

  c. (-100%) Running -> Waiting

  d. (33.33333%) Ready -> Waiting

Not possible state transition (Multiple choice)

- **Select the odd one out**

  a. (100%) Kernel stack of new process to kernel stack of scheduler

  b. (0%) Process stack of running process to kernel stack of running process

  c. (0%) Kernel stack of running process to kernel stack of scheduler

  d. (0%) Kernel stack of scheduler to kernel stack of new process

  e. (0%) Kernel stack of new process to Process stack of new process

Odd (stack transition) out (Multiple choice / One answer only)

- **The "push 0" in vectors.S is**

  a. (100%) Place for the error number value

  b. (0%) To be filled in as the return value of the system call

  c. (0%) A placeholder to match the size of struct trapframe

  d. (0%) To indicate that it's a system call and not a hardware interrupt

push 0 for errno (Multiple choice / One answer only)

- **The trapframe, in xv6, is built by the**

  a. (100%) hardware, vectors.S, trapasm.S

  b. (0%) vectors.S, trapasm.S

  c. (0%) hardware, vectors.S

  d. (0%) hardware, trapasm.S

  e. (0%) hardware, vectors.S, trapasm.S, trap()

Who builds trapframe? (Multiple choice / One answer only)

- **Calculate the EAT in NANO-seconds (upto 2 decimal points) w.r.t. a page fault, given Memory access time = {m} ns Average page fault service time = {t} ms Page fault rate = {p}**

Answer: (1-{p})*{m} + {p}*{t}*1000000

EAT (page fault) (Calculated)

- **Map the parts of a C code to the memory regions they are related to**
  - global initialized variables-> data
  - static variables-> data
  - global un-initialized variables-> bss
  - functions-> code
  - local variables-> stack
  - function arguments-> stack
  - malloced memory-> heap
  - -> buffers

C code parts, region mapping (Matching)

- **Suppose two processes share a library between them. The library consists of 5 pages, and these 5 pages are mapped to frames 9, 15, 23, 4, 7 respectively. Process P1 has got 6 pages, first 3 of which consist of process's own code/data and 3 correspond to library's pages 0, 2, 4. Process P2 has got 7 pages, first 3 of which consist of processe's own code/data and remaining 4 correspond to library's pages 0, 1, 3, 4. Fill in the blanks for page table entries of P1 and P2.**
  - Page table of P1, Page 3-> 9
  - Page table of P1, Page 4-> 23
  - Page table of P1, Page 5-> 7
  - Page table of P2, Page 0-> 9
  - Page table of P2, Page 1-> 15
  - Page table of P2, Page 3-> 4
  - Page table of P2, Page 4-> 7
  - -> 2
  - -> 3
  - -> 0
  - -> 5
  - -> 6

Find page table entries (Matching)

- **Map the technique with it's feature/problem**
  - static linking-> large executable file
  - dynamic linking-> small executable file
  - static loading-> wastage of physical memory
  - dynamic loading-> allocate memory only if needed

static/dynamic link/load (Matching)

- **Given below is the "maps" file for a particular instance of "vim.basic" process. Mark the given statements as True or False, w.r.t. the contents of the map file. 55a43501b000-55a435049000 r--p 00000000 103:05 917529 /usr/bin/vim.basic55a435049000-55a435248000 r-xp 0002e000 103:05 917529 /usr/bin/vim.basic55a435248000-55a4352b6000 r--p 0022d000 103:05 917529 /usr/bin/vim.basic55a4352b7000-55a4352c5000 r--p 0029b000 103:05 917529 /usr/bin/vim.basic55a4352c5000-55a4352e2000 rw-p 002a9000 103:05 917529 /usr/bin/vim.basic55a4352e2000-55a4352f0000 rw-p 00000000 00:00 0 55a436bc9000-55a436e5b000 rw-p 00000000 00:00 0 [heap]7f275b0a3000-7f275b0a6000 r--p 00000000 103:05 917901 /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so7f275b0a6000-7f275b0ad000 r-xp 00003000 103:05 917901 /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so7f275b0ad000-7f275b0af000 r--p 0000a000 103:05 917901 /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so7f275b0af000-7f275b0b0000 r--p 0000b000 103:05 917901 /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so7f275b0b0000-7f275b0b1000 rw-p 0000c000 103:05 917901 /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so7f275b0b1000-7f275b0b7000 rw-p 00000000 00:00 0 7f275b0b7000-7f275b8f5000 r--p 00000000 103:05 925247 /usr/lib/locale/locale-archive7f275b8f5000-7f275b8fa000 rw-p 00000000 00:00 0 7f275b8fa000-7f275b8fc000 r--p 00000000 103:05 924216 /usr/lib/x86_64-linux-gnu/libogg.so.0.8.47f275b8fc000-7f275b901000 r-xp 00002000 103:05 924216 /usr/lib/x86_64-linux-gnu/libogg.so.0.8.47f275b901000-7f275b904000 r--p 00007000 103:05 924216 /usr/lib/x86_64-linux-gnu/libogg.so.0.8.47f275b904000-7f275b905000 ---p 0000a000 103:05 924216 /usr/lib/x86_64-linux-gnu/libogg.so.0.8.47f275b905000-7f275b906000 r--p 0000a000 103:05 924216 /usr/lib/x86_64-linux-gnu/libogg.so.0.8.47f275b906000-7f275b907000 rw-p 0000b000 103:05 924216 /usr/lib/x86_64-linux-gnu/libogg.so.0.8.47f275b907000-7f275b90a000 r--p 00000000 103:05 924627 /usr/lib/x86_64-linux-gnu/libvorbis.so.0.4.87f275b90a000-7f275b921000 r-xp 00003000 103:05 924627 /usr/lib/x86_64-linux-gnu/libvorbis.so.0.4.87f275b921000-7f275b932000 r--p 0001a000 103:05 924627 /usr/lib/x86_64-linux-gnu/libvorbis.so.0.4.87f275b932000-7f275b933000 ---p 0002b000 103:05 924627 /usr/lib/x86_64-linux-gnu/libvorbis.so.0.4.87f275b933000-7f275b934000 r--p 0002b000 103:05 924627 /usr/lib/x86_64-linux-gnu/libvorbis.so.0.4.87f275b934000-7f275b935000 rw-p 0002c000 103:05 924627 /usr/lib/x86_64-linux-gnu/libvorbis.so.0.4.87f275b935000-7f275b937000 rw-p 00000000 00:00 0 7f275b937000-7f275b938000 r--p 00000000 103:05 917914 /usr/lib/x86_64-linux-gnu/libutil-2.31.so7f275b938000-7f275b939000 r-xp 00001000 103:05 917914 /usr/lib/x86_64-linux-gnu/libutil-2.31.so7f275b939000-7f275b93a000 r--p 00002000 103:05 917914 /usr/lib/x86_64-linux-gnu/libutil-2.31.so7f275b93a000-7f275b93b000 r--p 00002000 103:05 917914 /usr/lib/x86_64-linux-gnu/libutil-2.31.so7f275b93b000-7f275b93c000 rw-p 00003000 103:05 917914 /usr/lib/x86_64-linux-gnu/libutil-2.31.so7f275b93c000-7f275b93e000 r--p 00000000 103:05 915906 /usr/lib/x86_64-linux-gnu/libz.so.1.2.117f275b93e000-7f275b94f000 r-xp 00002000 103:05 915906 /usr/lib/x86_64-linux-gnu/libz.so.1.2.117f275b94f000-7f275b955000 r--p 00013000 103:05 915906 /usr/lib/x86_64-linux-gnu/libz.so.1.2.117f275b955000-7f275b956000 ---p 00019000 103:05 915906 /usr/lib/x86_64-linux-gnu/libz.so.1.2.117f275b956000-7f275b957000 r--p 00019000 103:05 915906 /usr/lib/x86_64-linux-gnu/libz.so.1.2.117f275b957000-7f275b958000 rw-p 0001a000 103:05 915906 /usr/lib/x86_64-linux-gnu/libz.so.1.2.117f275b958000-7f275b95c000 r--p 00000000 103:05 923645 /usr/lib/x86_64-linux-gnu/libexpat.so.1.6.117f275b95c000-7f275b978000 r-xp 00004000 103:05 923645 /usr/lib/x86_64-linux-gnu/libexpat.so.1.6.117f275b978000-7f275b982000 r--p 00020000 103:05 923645 /usr/lib/x86_64-linux-gnu/libexpat.so.1.6.117f275b982000-7f275b983000 ---p 0002a000 103:05 923645 /usr/lib/x86_64-linux-gnu/libexpat.so.1.6.117f275b983000-7f275b985000 r--p 0002a000 103:05 923645 /usr/lib/x86_64-linux-gnu/libexpat.so.1.6.117f275b985000-7f275b986000 rw-p 0002c000 103:05 923645 /usr/lib/x86_64-linux-gnu/libexpat.so.1.6.117f275b986000-7f275b988000 r--p 00000000 103:05 924057 /usr/lib/x86_64-linux-gnu/libltdl.so.7.3.17f275b988000-7f275b98d000 r-xp 00002000 103:05 924057 /usr/lib/x86_64-linux-gnu/libltdl.so.7.3.17f275b98d000-7f275b98f000 r--p 00007000 103:05 924057 /usr/lib/x86_64-linux-gnu/libltdl.so.7.3.17f275b98f000-7f275b990000 r--p 00008000 103:05 924057 /usr/lib/x86_64-linux-gnu/libltdl.so.7.3.17f275b990000-7f275b991000 rw-p 00009000 103:05 924057 /usr/lib/x86_64-linux-gnu/libltdl.so.7.3.17f275b991000-7f275b995000 r--p 00000000 103:05 921934 /usr/lib/x86_64-linux-gnu/libtdb.so.1.4.37f275b995000-7f275b9a3000 r-xp 00004000 103:05 921934 /usr/lib/x86_64-linux-gnu/libtdb.so.1.4.37f275b9a3000-7f275b9a9000 r--p 00012000 103:05 921934 /usr/lib/x86_64-linux-gnu/libtdb.so.1.4.37f275b9a9000-7f275b9aa000 r--p 00017000 103:05 921934 /usr/lib/x86_64-linux-gnu/libtdb.so.1.4.37f275b9aa000-7f275b9ab000 rw-p 00018000 103:05 921934 /usr/lib/x86_64-linux-gnu/libtdb.so.1.4.37f275b9ab000-7f275b9ad000 rw-p 00000000 00:00 0 7f275b9ad000-7f275b9af000 r--p 00000000 103:05 924631 /usr/lib/x86_64-linux-gnu/libvorbisfile.so.3.3.77f275b9af000-7f275b9b4000 r-xp 00002000 103:05 924631 /usr/lib/x86_64-linux-gnu/libvorbisfile.so.3.3.77f275b9b4000-7f275b9b5000 r--p 00007000 103:05 924631 /usr/lib/x86_64-linux-gnu/libvorbisfile.so.3.3.77f275b9b5000-7f275b9b6000 ---p 00008000 103:05 924631 /usr/lib/x86_64-linux-gnu/libvorbisfile.so.3.3.77f275b9b6000-7f275b9b7000 r--p 00008000 103:05 924631 /usr/lib/x86_64-linux-gnu/libvorbisfile.so.3.3.77f275b9b7000-7f275b9b8000 rw-p 00009000 103:05 924631 /usr/lib/x86_64-linux-gnu/libvorbisfile.so.3.3.77f275b9b8000-7f275b9ba000 r--p 00000000 103:05 924277 /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0.9.07f275b9ba000-7f275ba1e000 r-xp 00002000 103:05 924277 /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0.9.07f275ba1e000-7f275ba46000 r--p 00066000 103:05 924277 /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0.9.07f275ba46000-7f275ba47000 r--p 0008d000 103:05 924277 /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0.9.07f275ba47000-7f275ba48000 rw-p 0008e000 103:05 924277 /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0.9.07f275ba48000-7f275ba6d000 r--p 00000000 103:05 917893 /usr/lib/x86_64-linux-gnu/libc-2.31.so7f275ba6d000-7f275bbe5000 r-xp 00025000 103:05 917893 /usr/lib/x86_64-linux-gnu/libc-2.31.so7f275bbe5000-7f275bc2f000 r--p 0019d000 103:05 917893 /usr/lib/x86_64-linux-gnu/libc-2.31.so7f275bc2f000-7f275bc30000 ---p 001e7000 103:05 917893 /usr/lib/x86_64-linux-gnu/libc-2.31.so7f275bc30000-7f275bc33000 r--p 001e7000 103:05 917893 /usr/lib/x86_64-linux-gnu/libc-2.31.so7f275bc33000-7f275bc36000 rw-p 001ea000 103:05 917893 /usr/lib/x86_64-linux-gnu/libc-2.31.so7f275bc36000-7f275bc3a000 rw-p 00000000 00:00 0 7f275bc3a000-7f275bc41000 r--p 00000000 103:05 917906 /usr/lib/x86_64-linux-gnu/libpthread-2.31.so7f275bc41000-7f275bc52000 r-xp 00007000 103:05 917906 /usr/lib/x86_64-linux-gnu/libpthread-2.31.so7f275bc52000-7f275bc57000 r--p 00018000 103:05 917906 /usr/lib/x86_64-linux-gnu/libpthread-2.31.so7f275bc57000-7f275bc58000 r--p 0001c000 103:05 917906 /usr/lib/x86_64-linux-gnu/libpthread-2.31.so7f275bc58000-7f275bc59000 rw-p 0001d000 103:05 917906 /usr/lib/x86_64-linux-gnu/libpthread-2.31.so7f275bc59000-7f275bc5d000 rw-p 00000000 00:00 0 7f275bc5d000-7f275bcce000 r--p 00000000 103:05 917016 /usr/lib/x86_64-linux-gnu/libpython3.8.so.1.07f275bcce000-7f275bf29000 r-xp 00071000 103:05 917016 /usr/lib/x86_64-linux-gnu/libpython3.8.so.1.07f275bf29000-7f275c142000 r--p 002cc000 103:05 917016 /usr/lib/x86_64-linux-gnu/libpython3.8.so.1.07f275c142000-7f275c143000 ---p 004e5000 103:05 917016 /usr/lib/x86_64-linux-gnu/libpython3.8.so.1.07f275c143000-7f275c149000 r--p 004e5000 103:05 917016 /usr/lib/x86_64-linux-gnu/libpython3.8.so.1.07f275c149000-7f275c190000 rw-p 004eb000 103:05 917016 /usr/lib/x86_64-linux-gnu/libpython3.8.so.1.07f275c190000-7f275c1b3000 rw-p 00000000 00:00 0 7f275c1b3000-7f275c1b4000 r--p 00000000 103:05 917894 /usr/lib/x86_64-linux-gnu/libdl-2.31.so7f275c1b4000-7f275c1b6000 r-xp 00001000 103:05 917894 /usr/lib/x86_64-linux-gnu/libdl-2.31.so7f275c1b6000-7f275c1b7000 r--p 00003000 103:05 917894 /usr/lib/x86_64-linux-gnu/libdl-2.31.so7f275c1b7000-7f275c1b8000 r--p 00003000 103:05 917894 /usr/lib/x86_64-linux-gnu/libdl-2.31.so7f275c1b8000-7f275c1b9000 rw-p 00004000 103:05 917894 /usr/lib/x86_64-linux-gnu/libdl-2.31.so7f275c1b9000-7f275c1bb000 rw-p 00000000 00:00 0 7f275c1bb000-7f275c1c0000 r-xp 00000000 103:05 923815 /usr/lib/x86_64-linux-gnu/libgpm.so.27f275c1c0000-7f275c3bf000 ---p 00005000 103:05 923815 /usr/lib/x86_64-linux-gnu/libgpm.so.27f275c3bf000-7f275c3c0000 r--p 00004000 103:05 923815 /usr/lib/x86_64-linux-gnu/libgpm.so.27f275c3c0000-7f275c3c1000 rw-p 00005000 103:05 923815 /usr/lib/x86_64-linux-gnu/libgpm.so.27f275c3c1000-7f275c3c3000 r--p 00000000 103:05 923315 /usr/lib/x86_64-linux-gnu/libacl.so.1.1.22537f275c3c3000-7f275c3c8000 r-xp 00002000 103:05 923315 /usr/lib/x86_64-linux-gnu/libacl.so.1.1.22537f275c3c8000-7f275c3ca000 r--p 00007000 103:05 923315 /usr/lib/x86_64-linux-gnu/libacl.so.1.1.22537f275c3ca000-7f275c3cb000 r--p 00008000 103:05 923315 /usr/lib/x86_64-linux-gnu/libacl.so.1.1.22537f275c3cb000-7f275c3cc000 rw-p 00009000 103:05 923315 /usr/lib/x86_64-linux-gnu/libacl.so.1.1.22537f275c3cc000-7f275c3cf000 r--p 00000000 103:05 923446 /usr/lib/x86_64-linux-gnu/libcanberra.so.0.2.57f275c3cf000-7f275c3d9000 r-xp 00003000 103:05 923446 /usr/lib/x86_64-linux-gnu/libcanberra.so.0.2.57f275c3d9000-7f275c3dd000 r--p 0000d000 103:05 923446 /usr/lib/x86_64-linux-gnu/libcanberra.so.0.2.57f275c3dd000-7f275c3de000 r--p 00010000 103:05 923446 /usr/lib/x86_64-linux-gnu/libcanberra.so.0.2.57f275c3de000-7f275c3df000 rw-p 00011000 103:05 923446 /usr/lib/x86_64-linux-gnu/libcanberra.so.0.2.57f275c3df000-7f275c3e5000 r--p 00000000 103:05 924431 /usr/lib/x86_64-linux-gnu/libselinux.so.17f275c3e5000-7f275c3fe000 r-xp 00006000 103:05 924431 /usr/lib/x86_64-linux-gnu/libselinux.so.17f275c3fe000-7f275c405000 r--p 0001f000 103:05 924431 /usr/lib/x86_64-linux-gnu/libselinux.so.17f275c405000-7f275c406000 ---p 00026000 103:05 924431 /usr/lib/x86_64-linux-gnu/libselinux.so.17f275c406000-7f275c407000 r--p 00026000 103:05 924431 /usr/lib/x86_64-linux-gnu/libselinux.so.17f275c407000-7f275c408000 rw-p 00027000 103:05 924431 /usr/lib/x86_64-linux-gnu/libselinux.so.17f275c408000-7f275c40a000 rw-p 00000000 00:00 0 7f275c40a000-7f275c418000 r--p 00000000 103:05 924540 /usr/lib/x86_64-linux-gnu/libtinfo.so.6.27f275c418000-7f275c427000 r-xp 0000e000 103:05 924540 /usr/lib/x86_64-linux-gnu/libtinfo.so.6.27f275c427000-7f275c435000 r--p 0001d000 103:05 924540 /usr/lib/x86_64-linux-gnu/libtinfo.so.6.27f275c435000-7f275c439000 r--p 0002a000 103:05 924540 /usr/lib/x86_64-linux-gnu/libtinfo.so.6.27f275c439000-7f275c43a000 rw-p 0002e000 103:05 924540 /usr/lib/x86_64-linux-gnu/libtinfo.so.6.27f275c43a000-7f275c449000 r--p 00000000 103:05 917895 /usr/lib/x86_64-linux-gnu/libm-2.31.so7f275c449000-7f275c4f0000 r-xp 0000f000 103:05 917895 /usr/lib/x86_64-linux-gnu/libm-2.31.so7f275c4f0000-7f275c587000 r--p 000b6000 103:05 917895 /usr/lib/x86_64-linux-gnu/libm-2.31.so7f275c587000-7f275c588000 r--p 0014c000 103:05 917895 /usr/lib/x86_64-linux-gnu/libm-2.31.so7f275c588000-7f275c589000 rw-p 0014d000 103:05 917895 /usr/lib/x86_64-linux-gnu/libm-2.31.so7f275c589000-7f275c58b000 rw-p 00000000 00:00 0 7f275c5ae000-7f275c5af000 r--p 00000000 103:05 917889 /usr/lib/x86_64-linux-gnu/ld-2.31.so7f275c5af000-7f275c5d2000 r-xp 00001000 103:05 917889 /usr/lib/x86_64-linux-gnu/ld-2.31.so7f275c5d2000-7f275c5da000 r--p 00024000 103:05 917889 /usr/lib/x86_64-linux-gnu/ld-2.31.so7f275c5db000-7f275c5dc000 r--p 0002c000 103:05 917889 /usr/lib/x86_64-linux-gnu/ld-2.31.so7f275c5dc000-7f275c5dd000 rw-p 0002d000 103:05 917889 /usr/lib/x86_64-linux-gnu/ld-2.31.so7f275c5dd000-7f275c5de000 rw-p 00000000 00:00 0 7ffd22d2f000-7ffd22d50000 rw-p 00000000 00:00 0 [stack]7ffd22db0000-7ffd22db4000 r--p 00000000 00:00 0 [vvar]7ffd22db4000-7ffd22db6000 r-xp 00000000 00:00 0 [vdso]ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0 [vsyscall]**

Meaning of "maps" file (Multiple True False (ETH))

- **W.r.t the figure given below, mark the given statements as True or False.**

page replacement choices (Multiple True False (ETH))

- **Select all the correct statements, w.r.t. Copy on Write**

  a. (25%) Fork() used COW technique to improve performance of new process creation.
  b. (25%) If either parent or child modifies a COW-page, then a copy of the page is made and page table entry is updated
  c. (25%) COW helps us save memory
  d. (25%) Vfork() assumes that there will be no write, but rather exec()
  e. (-50%) use of COW during fork() is useless if exec() is called by the child
  f. (-50%) use of COW during fork() is useless if child called exit()

COW T/F (Multiple choice)

- **Given below is the output of the command "ps -eo min_flt,maj_flt,cmd" on a Linux Desktop system. Select the statements that are consistent with the output 626729 482768 /usr/lib/firefox/firefox -contentproc -parentBuildID 20220202182137 -prefsLen 9256 -prefMapSize 264738 -appDir /usr/lib/firefox/browser 6094 true rdd 2167 687 /usr/sbin/apache2 -k start 1265185 222 /usr/bin/gnome-shell 102648 111 /usr/sbin/mysqld 9813 0 bash 15497 370 /usr/bin/gedit --gapplication-service**

  a. (25%) Firefox has likely been running for a large amount of time
  b. (25%) Apache web-server has not been used much
  c. (25%) The bash shell is mostly busy doing work within a particular locality
  d. (25%) All of the processes here exihibit some good locality of reference

meaning of ps -eo min_flt, etc. (Multiple choice)

- **which of the following, do you think, are valid concerns for making the kernel pageable?**

  a. (25%) The kernel's own page tables should not be pageable
  b. (25%) The page fault handler should not be pageable
  c. (25%) The kernel must have some dedicated frames for it's own work
  d. (25%) The disk driver and disk interrupt handler should not be pageable
  e. (-50%) No data structure of kernel should be pageable
  f. (-50%) No part of kernel code should be pageable.

pageable kernel (Multiple choice)

- **Order the following events, related to page fault handling, in correct order**
  - MMU detects that a page table entry is marked "invalid"
  - Page fault interrupt is generated
  - Page fault handler in kernel starts executing
  - Page fault handler detects that it's a page fault and not illegal memory access
  - Empty frame is found
  - Disk read is issued
  - Page faulting process is made to wait in a queue
  - Other processes scheduled by scheduler
  - Disk Interrupt occurs
  - Disk interrupt handler runs
  - Page table of page faulted process is updated
  - Page faulted process is moved to ready-queue

Demand paging : order (Ordering)

- **Assuming a 8- KB page size, what is the page numbers for the address {address} reference in decimal : (give answer also in decimal)**

Answer: {address} / (8*1024)

page number calculation (Calculated)

- **Given six memory partitions of 300 KB , 600 KB , 350 KB , 200 KB , 750 KB , and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB and 500 KB (in order)?**
  - first fit 115 KB

    -> 300 KB
  - first fit 500 KB

    -> 600 KB
  - best fit 115 KB

    -> 125 KB
  - best fit 500 KB

    -> 600 KB
  - worst fit 115 KB

    -> 750 KB
  - worst fit 500 KB

  - -> 635 KB
  - -> 200 KB
  - -> 350 KB

first/worst/best fit (Matching)

- **For the reference string 3 4 3 5 2 the number of page faults (including initial ones) using FIFO replacement and 2 page frames is : {#1} FIFO replacement and 3 page frames is : {#2}**
  - {1:SHORTANSWER:%100%4}
  - {1:SHORTANSWER:%100%4}

#page faults (Embedded answers (Cloze))

- **Page sizes are a power of 2 because**

  a. (100%) Certain bits are reserved for offset in logical address. Hence page size = 2^(no.of offset bits)
  b. (0%) operating system calculations happen using power of 2
  c. (0%) MMU only understands numbers that are power of 2
  d. (0%) Power of 2 calculations are highly efficient
  e. (0%) Certain bits are reserved for offset in logical address. Hence page size = 2^(32 - no.of offset bits)

Page sizes are a power of 2 because (Multiple choice / One answer only)

- **Compare paging with demand paging and select the correct statements.**

  a. (14.28571%) Demand paging requires additional hardware support, compared to paging.
  b. (14.28571%) Both demand paging and paging support shared memory pages.
  c. (14.28571%) With demand paging, it's possible to have user programs bigger than physical memory.
  d. (14.28571%) Demand paging always increases effective memory access time.
  e. (14.28571%) Paging requires some hardware support in CPU
  f. (14.28571%) Calculations of number of bits for page number and offset are same in paging and demand paging.
  g. (14.28571%) The meaning of valid-invalid bit in page table is different in paging and demand-paging.
  h. (-33.33333%) With paging, it's possible to have user programs bigger than physical memory.
  i. (-33.33333%) Paging requires NO hardware support in CPU
  j. (-33.33333%) TLB hit ration has zero impact in effective memory access time in demand paging.

paging vs demand paging (Multiple choice)

- **Shared memory is possible with which of the following memory management schemes ?**

  a. (33.33333%) paging
  b. (33.33333%) segmentation
  c. (-100%) continuous memory management
  d. (33.33333%) demand paging

shared memory - possible (Multiple choice)

- **Arrange the following in the correct order of execution (w.r.t. 'init')**
  - userinit() is called-> 1
  - 'initcode' struct proc is created-> 2
  - 'initcode' process is marked RUNNABLE-> 3
  - mpmain() calls scheduler()-> 4
  - scheduler() schedules initcode() process-> 5
  - initcode() returns in forkret()-> 6
  - initcode() returns from trapret()-> 7
  - initcode() calls exec("/init", ...)-> 8

init related execution sequence (Matching)

- **Map the virtual address to physical address in xv6**
  - KERNBASE-> 0
  - KERNLINK-> 0x100000
  - 80108000-> 0x108000
  - 0xFE000000-> 0xFE000000
  - -> 0x80000000

kernel memory mappings (Matching)

- **The approximate number of page frames created by kinit1 is**

a. (100%) 3000
b. (0%) 1000
c. (0%) 2000
d. (0%) 4000
e. (0%) 10
f. (0%) 4
g. (0%) 16

#kinit1's pages (Multiple choice / One answer only)

- **Select all the correct statements about initcode**

a. (25%) code of 'initcode' is loaded along with the kernel during booting
b. (25%) the size of 'initcode' is 2c
c. (25%) The data and stack of initcode is mapped to one single page in userinit()
d. (25%) initcode essentially calls exec("/init",...)
e. (-33.33333%) initcode is the 'init' process
f. (-33.33333%) code of initcode is loaded in memory by the kernel during userinit()
g. (-33.33333%) code of initcode is loaded at virtual address 0

correct about initcode (Multiple choice)

- **Which of the following is DONE by allocproc() ?**

a. (20%) Select an UNUSED struct proc for use
b. (20%) allocate PID to the process
c. (20%) allocate kernel stack for the process
d. (20%) setup the trapframe and context pointers appropriately
e. (20%) ensure that the process starts in forkret()
f. (-33.33333%) ensure that the process starts in trapret()
g. (-33.33333%) setup kernel memory mappings for the process
h. (-33.33333%) setup the contents of the trapframe of the process properly

not done by allocproc() (Multiple choice)

- **Which of the following is done by mappages()?**

a. (33.33333%) create page table mappings for the range given by "va" and "va + size"
b. (33.33333%) allocate page table if required
c. (33.33333%) create page table mappings to the range given by "pa" and "pa + size"
d. (-50%) allocate page directory if required
e. (-50%) allocate page frame if required

not done by mappages (Multiple choice)

- **What does seginit() do?**

a. (100%) Adds two additional entries to GDT corresponding to Code and Data segments, but to be used in privilege level 3
b. (0%) Adds two additional entries to GDT corresponding to Code and Data segments, but to be used in privilege level 0
c. (0%) Nothing significant, just repetition of earlier GDT setup but with 2-level paging setup done
d. (0%) Nothing significant, just repetition of earlier GDT setup but with free frames list created now
e. (0%) Nothing significant, just repetition of earlier GDT setup but with kernel page table allocated now

seginit() does? (Multiple choice / One answer only)

- **Select the statement that most correctly describes what setupkvm() does**

a. (100%) creates a 2-level page table setup with virtual->physical mappings specified in the kmap[] global arrray
b. (0%) creates a 2-level page table setup with virtual->physical mappings specified in the kmap[] global arrray and makes kpgdir point to it
c. (0%) creates a 2-level page table for the use of the kernel, as specified in gdtdesc
d. (0%) creates a 1-level page table for the use by the kernel, as specified in kmap[] global array

setupkvm()'s job (Multiple choice / One answer only)

- **What does userinit() do ?**

a. (100%) sets up the 'initcode' process to start execution in forkret()
b. (0%) sets up the 'init' process to start execution in forkret()
c. (0%) sets up the 'initcode' process to start execution in trapret()
d. (0%) sets up the 'initcode' process to start execution in forkret ()
e. (0%) initializes the users
f. (0%) initializes the process 'init' and starts executing it

userinit() does? (Multiple choice / One answer only)

- **The variable 'end' used as argument to kinit1 has the value**

a. (100%) 801154a8
b. (0%) 80110000
c. (0%) 80000000
d. (0%) 81000000
e. (0%) 80102da0
f. (0%) 8010a48c

value of end (Multiple choice / One answer only)

- **Does exec() code around clearptau() lead to wastage of one page frame?**

a. (100%) yes
b. (0%) no

wastage in exec? (Multiple choice / One answer only)

- **exec() does this: curproc->tf->eip = elf.entry, but userinit() does this: p->tf->eip = 0; Select all the statements from below, that collectively explain this**

a. (33.33333%) exec() loads from ELF file and the address of first instruction to be executed is given by 'entry'
b. (33.33333%) In userinit() the function inituvm() has mapped the code of 'initcode' to be starting at virtual address 0
c. (33.33333%) the initcode is created using objcopy, which discards all relocation information and symbols (like entry)
d. (-33.33333%) the 'entry' in initcode is anyways 0
e. (-33.33333%) the code of 'initcode' is loaded at physical address 0
f. (-33.33333%) elf.entry is anyways 0, so both statements mean the same

why different eip settings? (Multiple choice)

- **Why is there a call to kinit2? Why is it not merged with knit1?**

a. (100%) knit2 refers to virtual addresses beyond 4MB, which are not mapped before kvalloc() is called
b. (0%) Because there is a limit on the values that the argumets to knit1() can take.
c. (0%) When kinit1() is called there is a need for few page frames, but later knit2() is called to serve need of more page frames
d. (0%) call to seginit() makes it possible to actually use PHYSTOP in argument to kinit2()

why kinit2()? (Multiple choice / One answer only)
