- **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)
