Harry Langford
hjel2@cam.ac.uk

Operating Systems SV3

File Systems and IO

The data is accessed in order of easiest to hardest. This means that data goes from immediate data -> direct block pointers -> indirect block pointer -> double indirect block pointer. The order is not necessarily from top to bottom or bottom to top.

Indirect block pointers reference new files. So you can access the 0th -> 1024 * 4kbth bytes via an indirect block pointer.

In the sample question: there are 2040 + 256 * 4096 + 1024 * 4096 = $\approx 5Mb$.

Note that indirect block pointers point to disk blocks which are full of direct block pointers.

Text, Data and Stack Segments:

The text segment changes *only* when loaded in. It does not change during user execution.

The data segment has subsegments: the heap is a subsegment of the data segment as is the part which has static / global data section.

What data is put on the heap depends on what language you use. In C all local variables are in the stack and to allocate something on the heap you use malloc, calloc and free. Other languges abstract this away further so C++ when you create new objects then those are created on the heap. Java has its own memory management. Java has a call stack and a heap. Everything that's an object goes on the heap (all primitive types go onto the stack).

The Stack sits on top of the inode and contains stack frames. Each stackc frame represents a function which has been called. The stack contains the return address, arguments, local variables and point of execution. This means that when the function complees you know where to return. Upon calling functions, you allocate new stack frames. So the stack "grows" downwards. When the stack semgent gets full because you have too many stack frames, you just increase the size of the stack segment.

Each process also has a kernel space part. This is a copy of kernel memory.

IO:

There are two separate interfaces when it comes to IO: There is the operating system and the user process. These two communicate via system calls. This is one interface.

As part of the operating system we've got device drivers. They communicate with the hardware.

Polled/Interrupt driven is between the device driver and the I/O device.

The IO types are between the user process and the system call.

Blocking IO blocks the program until all the data is there. So when the process continues it knows that all the data is there.

Nonblocking IO will return all the data that is immediately available. The "polling" part is that you can do repeated Nonblocking IO system calls until there is enough. Nonblocking is not polling. You say "give me all you've got right now". And it does.

Device Drivers:

You issue requests to the drive when you want to read or write to files from the hard disk or evict a page onto secondary storage.

Say we issue a read request. This takes time. When this finishes we put an interrupt on the hard disk so that the CPu knows the data is now ready.

At the end of an ISR we have to call the scheduler – what if a process has gone from blocked into the ready state?

Given that the hard-disk drive is not really a random access device, what steps could you take to improve performance?

Reduce the seek time by pooling instructions and scheduling. This is most useful for writing since ther eis a write buffer where you can hold dirty pages and write back intermittently.

Locality of Reference:

You deal in pages, you pol in the whole page. This means that you ahve all the bytes which are near to the byte that you've just accessed. This means that you have all the bytes you might need. The page table entry gives you the translation for all the nearby addresses. This means that you have all the translation information for all nearby addresses if you load in one address.

Buses:

In asynchronous buses you don't have to pick the clock speed of the slowest device. This means that ie the CPU is not limited by the speed of the memory.

For a single bus you've got a master-slave setup. This means that the master is the only one which can initiate transactions on the bus.

There is another version which allows multiple masters (any device ont eh bus can initiate the transaction). This is useful for things such as DMA. This allows devices to share buses.