SOLUTION NOTES

Operating System Foundations 2002 Paper 11 Question 5 (GMB)

(*a*) (*i*) **Interrupt-based I/O** [Slide 99] CPU uses bus to make a request. The device goes off to perform operation, meanwhile the CPU continues doing other work. When the IO operation has completed the device raises an interrupt. When an interrupt is raised, then the CPU saves the PC and other registers etc., changes processor mode and then vectors to the handler. The handler would then handle the IO transfer. After the interrupt handling routine is completed the original program is resumed (using some special instruction, e.g. `rti`). This will involve reloading the registers etc.

(*ii*) **Direct Memory Access** [Slide 31] Often there is a special-purpose processor called a DMA-controller to handle IO. To initiate a DMA transfer, the host writes a DMA command to memory. This typically contains the source address, the destination address, and the size of the transfer (possibly also an indication of how the successive addresses for source and destination are to be calculated, e.g. inc, dec, nop). The CPU then writes the address of this command to the DMA controller, and then gets on with other work. The DMA then operates the memory bus directly and manages the transfer without the help of the main CPU. When the entire transfer is finished the DMA controller interrupts the CPU.

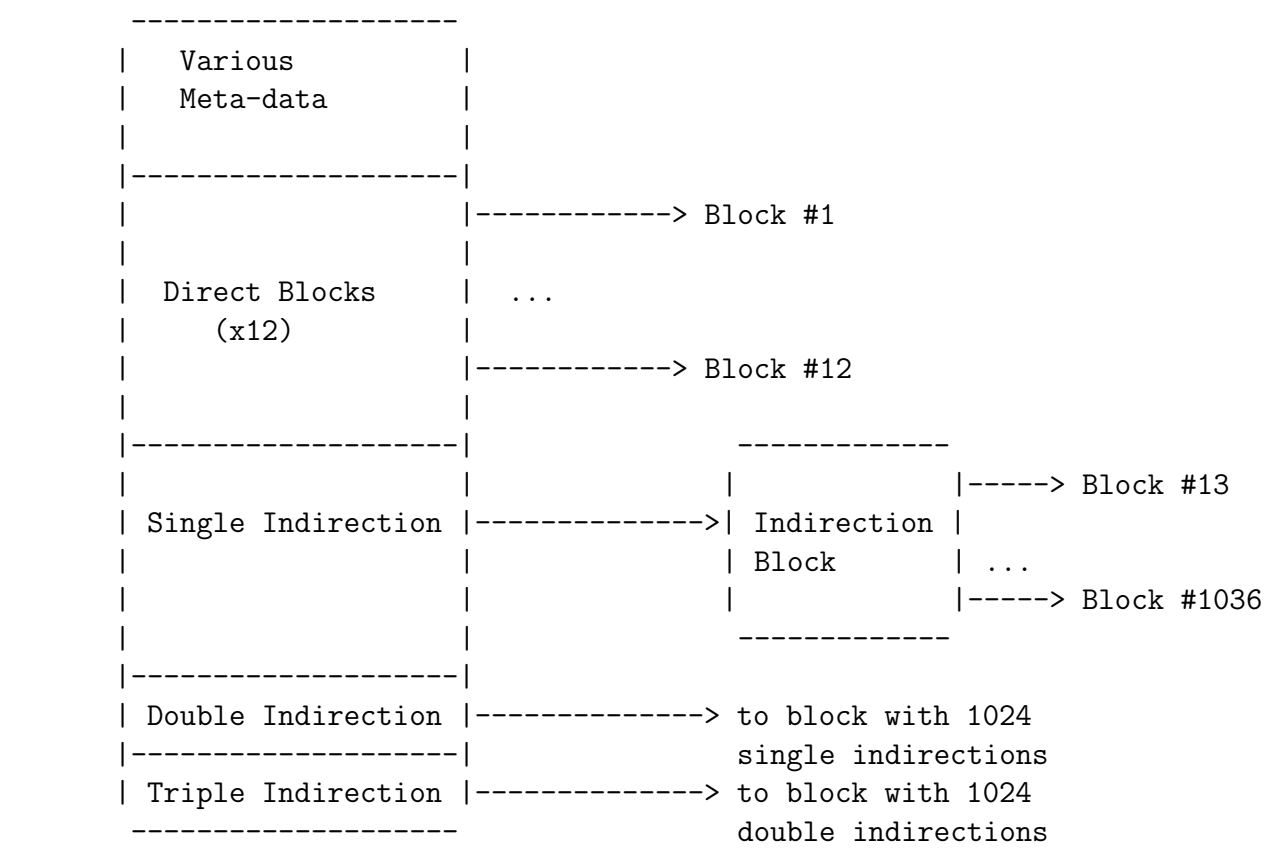
(*iii*) **File access control in Unix** [Slide 128] Access control information is held in each inode. There are three bits – read, write, and execute – for each of owner, group and world. For example `0640` mean RW for owner, R for group and nothing for world.

In addition there is a setuid/setgid bit which allows a user to become someone else when running a given program. In UNIX every process has a real UID and an effective UID. If there exists a file, `test` owned by `gmb` with E set for world/group and also SETUID set, it means that if some other user, `britney`, executes the file `test` their effective UID is set to `gmb`. For example this program could update a file `scores` that was only accessible to owner `gmb` and not to `britney`.

(*iv*) **Mounting file systems in Unix** [Slide 126] A file system must be mounted before it is available to processes on a system. More specifically, the directory structure can be built out of multiple partitions, which must be mounted to make them available within the file systems namespace. The operation is fairly straightforward; the OS is given the name of the file system to be mounted, and the location within the file system at which to attach it, e.g. `mount("/dev/hda2","/home/britney",...)`. This hence provides a unified namespace (e.g. if there was a file `/song/baby.txt` in the filesystem on `/dev/hda2`, then it will be available as `/home/britney/song/baby.txt` after the mount.

UNIX does not allow hard links across mount points, but does allow soft links.

(*b*) (*i*) **UNIX inode structure** [Slide 123] The Unix (Version 7) inode structure is as follows.



Meta-data stored typically includes the user ID, group ID, mode, timestamp (create, last updated, last read), reference count (number of hard links from directories - if gets down to zero, then file can be deleted and inode garbage collected). Note the filename is not stored in the inode, but in the directory.

(*ii*) The maximum file size is then 12+1024+(1024*1024)+(1024*1024*1024) blocks.