Operating systems – supervision 3

James Wood

March 12, 2015

1. (a) Virtual addresses mean that processes have no way of accessing memory outside their address space, the OS can change the physical location of memory without the program needing to know about it, and virtual memory can be larger than physical memory if extra storage (such as a hard disk) is used.

   (b) Virtual addressing requires additional hardware to be performant. For embedded systems, the cost (by whatever metric) of the extra parts (and the time taken to translate each address) may not be worth it, since programs may have physical addresses bound at compile or load time.

   (c) i. In both cases, applications get access to addresses starting at 0. In the case of segmented memory, these addresses are bounded, but with paged memory, these are unbounded (though the system may crash in the event that too much memory is used). For segmented memory, there will be a simple function mapping virtual addresses to physical addresses based on what range the virtual address is in. Here is an example:

virtual address	physical address
[0..200)	[7800..8000)
[200..600)	[400..800)
[600..E00)	[4800..5000)

For paged memory, virtual addresses are assigned to pages, then pages are assigned to physical addresses. Here is an example state of this system:

virtual address	page number	physical address
[0..1000)	5	[24000..25000)
[1000..2000)	93	[8000..9000)
[2000..3000)	34	[1A000..1B000)

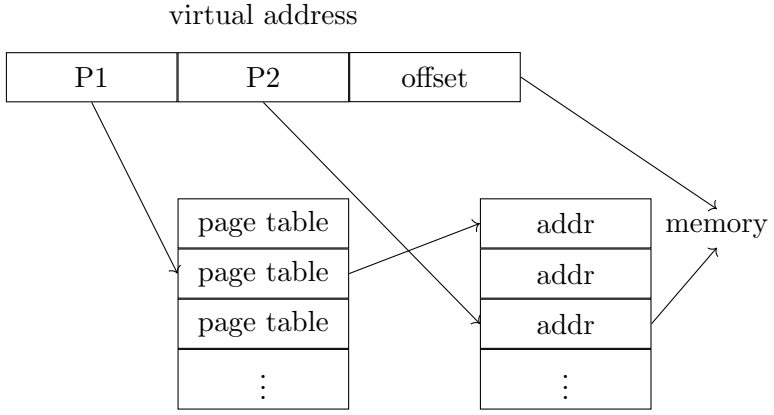
ii. Memory allocation is easier for paged memory because finding free space requires only a lookup in the page table. With segmented memory, each PCB must be checked, and the OS has to reason about ranges.

iii. Segmented memory is more reliably quick to access, since it doesn't rely on much indirection. It also doesn't require as much hardware support, since paging is usually supplemented with a TLB.

2. (a) Internal fragmentation occurs when a process has been allocated more memory than it needs. This could occur because of a memory leak, lack of recent garbage collection, or a sudden decrease in memory requirements of the process. This can be mitigated by giving processes that have growing memory requirements a lower priority in the scheduler. External fragmentation occurs in segmenting systems when the physical memory of processes is arranged in such a way that there are small gaps between the segments of different processes. This can occur when processes change their memory demands, or when a small process terminates. This problem can be solved by periodically reördering memory, assuming that processes only have access to memory via virtual addresses.

   (b) The page table gives a comprehensive mapping from processes' virtual addresses to physical addresses. The TLB contains some of these mappings – particularly mappings that are likely to be needed soon (like mappings associated with the current process). When there is a change in the page table, the TLB should be notified so that it doesn't become invalid.

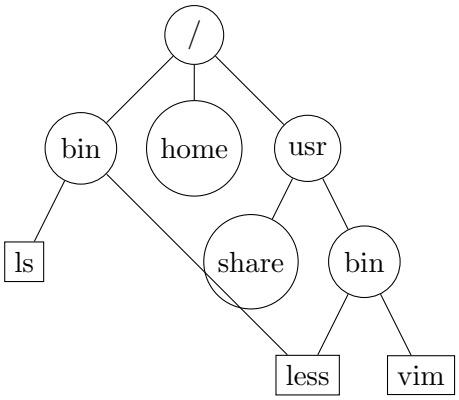
   (c)



Multilevel page tables are useful when the computer has a very large address space, giving it many pages. However, most of these usually go unused, so it represents a large saving in memory to only create page tables when they are needed.

3. (a) The directory service gives files a hierarchical structure and manages permissions.

(b)



A directory hierarchy can be represented as a directed graph. For the convenience of garbage collection routines, it is usually required to by acyclic, which can be enforced by requiring directories to have at most one parent. In this system, the directories form a tree, but can link to non-directories arbitrarily.

(c) File metadata may store an ID or name for the file, the owner and group of the file, a count of references to the file, and dates of creation and modification.

(d) Hard linking is the way of making a file a member of a directory. A hard link is guaranteed to point to a file, and when the last hard link to a file is deleted, the space the file was using can be freed.

(e) A soft link is a file that points to another file. It is not guaranteed to point to anything, and will cause an exception if opened while not pointing to anything.

4. Unix gives each file a set of permissions. The permissions pertain separately to the owner, users in the same group as the owner, and others, and allow them to have any of read, write and execute access. More complex access control policies can be approximated by allowing files to be executed as a user other than the user who is actually using the program. The command `su` is one program offering this functionality.

(a)

boot block	super block	inodes	data blocks...

The boot block is reserved for data needed in the booting process. The super block contains information about the file system, including its size and number of inodes. It also contains a pointer to a list of free data blocks. This helps with portability, since values in the super block (rather than many pieces of source code) are all that have to be changed when porting to a different file system.

(b) See question 6.

(c) With a block size of 512 B, 10 direct blocks, and 4 B pointers, each indirect block has 128 pointers, giving a total maximum file size of $512\,\mathrm{B} \cdot (10 + 128 + 128^2 + 128^3)$, which is just over 1 GiB.

(d)
- The Unix V7 file system suffers from the fact that accessing a file requires at least two disk reads which are likely to be in completely different places. To fix this, inodes should be stored at the same radius as the data blocks associated with them. There should also be multiple super blocks, providing multiple lists of free blocks.
- Also, at 512 B, blocks are too small. It saves on disk reads to have large blocks, even if it requires greater disk useage.
- Reducing the number of disk reads will also improve reliability, since there is less chance for a disk access to go wrong.

5. It needs to be impossible for users to modify the archived files in-place, so all write permissions should be removed. However, whole archived roots should be deletable by the administrator, so (only) these directories need write permissions for the owner, with the owner being the administrator.

   It is to be noted that directories can have at most one hard link pointing to them, so in the copy process, all directories must actually be copied. However, files need not be copied until they are modified.

6.

mode (directory or regular file)
device ID, user ID, group ID
size
atime, mtime, ctime
link count
direct blocks
single indirect block
double indirect block
triple indirect block

   atime is the time of the last access to the file. mtime is the time of the last modification of the file's contents. ctime is the time of the last change of the inode or its contents. A link count (a count of hard links referencing the inode) is maintained for garbage collection. The blocks section contains an array of pointers to data blocks. Because this is of fixed size, the indirect blocks are provided. The single indirect block points to an array of pointers to data blocks, whilst the double indirect block points to an array of arrays and the triple indirect block points to an array of arrays of arrays.

7.

8. (a) There is a delay after entering an incorrect password, and passwords are never printed to the screen.

   (b) The password file contains only hashes of passwords, not passwords themselves, and passwords are salted before being hashed to avoid dictionary attacks.

9. • CPU

- Main memory
- Hard drive
- Monitor
- Keyboard

10. (a) Access control lists are stored with the files they pertain to, whilst capabilities are stored with the user. With the capabilities model, it is easy to add new users with few permissions, and all such users can be given access to some shared files in an automatic way. When using ACLs, users generally have to be arranged into groups so that files can be created without needing to know exactly what users there are on the system. Deciding accessibility for capabilities is simpler, since it requires only checking a key (which can be done in hardware).

    (b)

    (c) Capabilities can be emulated in a POSIX system using groups and setuid. In theory, a user could be made for each possible set of permissions an administrator could want other users to have at various times, then allow them to be that user for the required action.

    (d) Capability-based systems allow finer control of what each user can do to each file, so it is possible to get closer to the ideal of users having only the permissions they need. This improves security because it limits what users could (possibly accidentally) do, reducing the chance of the system becoming compromised.

11. - If the size of the problem is secret and the algorithm's performance varies with the size of the problem, the algorithm could be timed to find out the secret. This can be mitigated by using a different algorithm.