

COS214 Tutorial 7

Roberto Togneri, 2000

SOLUTIONS

1. (a) The Macintosh OS allows file names of the form “Tutorial Six”, so does MS Windows 95. As the C shell with the UNIX OS uses a space as a command line separator what are the consequences of having a name with an embedded space? Could you have a UNIX filename with an end of line character in it for example?
- (b) A company uses a mixture of UNIX workstations and Pentium PC’s running Windows 95 connected to the same network. For ease of administration user accounts are stored on a UNIX file server and all machines have access to the server across the network. A user, on one of the UNIX workstations, creates a file and names it, *ChocolateChipCookies.doc*. The same user now wishes to edit that file on a Pentium PC using MS Word. Can this be done? If not, can you suggest a cure? Does it make any difference if the Pentium is running Windows 3.1x or Windows 95? What if the file was created on one of the Pentium systems and then accessed via a UNIX workstation?

Answer

- (a) In UNIX files with non-printable (^L, ^M, ^V, etc) and non-alphanumeric (‘,!,?,;) can be created by specifying the ASCII code directly in the filename string. (usually via a C system call like create()) or, depending on the shell, specify the names enclosed by ‘ ’ or “ ”. Such files are hard to manipulate and especially remove.
- (b) Win 3.1 only supports 8.3 file names. Win 95 will be okay as it support long file names BUT the UNIX server will need to support the format conversion since UNIX and Win 95 support long file names in different ways. The problem of accessing a file created on a Win 95 system from a UNIX workstation is less problematic since UNIX has greater filename flexibility (except for non-printable characters and different case conversion semantics).

2. (a) What is the purpose of a filename extension? How does UNIX know whether a file contains an executable program?
- (b) UNIX has many different file types. What is the purpose of this? Give examples of the different types.
- (c) List the typical attributes a files might have? Can you think of other attributes a file ought to have?

Answer

- (a) File name extensions are basically useful to the user. In the case of Windows 3.x and 95 filename extensions are used to associate documents with programs. Under UNIX the magic number may be used to identify executable/text/binary etc file types, although this is not rigorously the case.

- (b) Flexibility for process to process and process to peripheral interaction:

regular files (-)
directories (d)
character (serial I/O, raw disk, tape device) (c)
block (block I/O) (b)
symbolic links (l)
sockets (s)
fifo (p)

- (c) Typical and other attributes:

filename
read, write, execute access.
setuid, setgid, for execute ownership
filesize
access/create/modify times
hard links
type of file
read-only, hidden, system, archive (MS-DOS)
icon, working directory, minimised (MS-WINDOWS)

3. (a) If */usr/jim* is the working directory, what is the absolute path name for the file whose relative path name is *../ast/x*?
- (b) What are the advantages and disadvantages of recording the name of the creating program with the file’s attributes (as is done in the Macintosh operating system)?

Answer

- (a) */usr/ast/x*
- (b) By recording the name of the creating program, the operating system is able to implement features (such as automatic program invocation when the file is accessed) based on this information. It does add overhead in the operating system and require space in the file descriptor, however. Furthermore, the creating program may have been removed, modified or may simply not be the program you want to use on the file.

4. (a) A file can be moved in two different ways:
- (i) Simply “rename” the file by updating the directory information
- (ii) Copying the file to the new location and deleting the original file
- Compare the two approaches. What are the corresponding UNIX commands?
- (b) Some systems automatically open a file when it is referenced (for reading or writing) for the first time, and close the file when the process terminates. Discuss the advantages and disadvantages of this scheme as compared to the more traditional one, where the user has to open and close the file explicitly.

Answer

- (a) Renaming the file is faster than copying and deleting data, especially for large files. However it is usually limited to work within same filesystem and cannot be used to copy files across different filesystems or media. Under UNIX the *mv* command implements the rename operation (if on the same filesystem), *cp* implements the file copy and *rm* implements the file remove.

(b) Automatic opening and closing of files relieves the user from the invocation of these functions, and thus makes it more convenient for the user; however, it requires more overhead than the case where explicit opening and closing is required. Furthermore, it removes needed flexibility: What happens if the user needs to open a file in order to get the file handler (to pass to another routine, child process or thread) but does not otherwise to read or write to the file? What happens if the process first referencing the file passes the handler to a thread or child process to continuing processing and then terminates (and automatically closes the file even though the child process or thread is still accessing the data)?

5. (a) An operating system only supports a single directory but allows that directory to have arbitrarily many files with arbitrarily long file names. Can something approximating a hierarchical file system be simulated? How?
- (b) Directories can be implemented as “special files” that can only be accessed in limited ways, or as ordinary data files. What are the advantages and disadvantages of each approach?

Answer

(a) Use filenames such as `/usr/ast/file`. While it looks like a hierarchical path name, it is really just a single name containing embedded slashes.

(b) Directories should be special files in that directory operations are different than file operations (e.g. you can't simply delete a directory file without first deleting all the entries), so the operating system should implement directories differently to ordinary files. The only disadvantage is the extra processing and data structures to support an additional file type.

6. (a) The UNIX file system supports hard and symbolic links. Is there an equivalent technique in MS Windows 95? Does this technique behave like a hard link or symbolic link?
- (b) What additional parameters should be passed to the file create function besides the name of the file?
- (c) It has been suggested that the first part of each Unix file be kept in the same disk block as its inode, rather than locating the inodes in the first part of the filesystem. What would the advantages be? And what are the disadvantages?

Answer

(a) Windows 95 supports “shortcuts” which are actually extra files stored on the disk containing information about what they point. Shortcuts act as symbolic links.

(b) `create(filename, type of file, access permissions)`

(c) *Advantage:* avoid long seeks as data and i-nodes are close; no limitation on the number of files that can exist
Disadvantages: no good for random access of larger files or accessing all of a large file; i-node entries are no longer together on the same area of disk making it longer to search for the location of i-node itself, especially if its placement is random.