Computer File Management

John C Nash, Telfer School of Management, University of Ottawa

2023-01-20

Contents

Preface

This tutorial is about the housekeeping tasks which allow data to be managed properly on computer systems. Data could be textual, graphical or numeric material; moreover, it may be stored locally in a form directly connected to the computer (main storage), in collections of removable storage media or on remote networked storage. As with all housekeeping, it concerns the unglamorous jobs necessary for a healthy and viable operation. Much of the work needed to maintain systems of files is tedious and somewhat boring. Failure to maintain the files is less boring but usually has unpleasant consequences.

In essence, we are talking about taking out the (computer) garbage. Just as no house is livable for more than a couple of weeks unless the kitchen refuse is removed, no computer system can be used effectively unless the waste data is eliminated. Similarly, in our domestic lives we organize our belongings for convenience, arranging things in cupboards and drawers so they can be kept neat and clean and out of our way. In computers, we must likewise organize data for efficient use.

Much has been written on the creation and structuring of computer files, especially under the rubric of 'databases' but very little on their archiving and practically nothing on their proper disposal. The life cycle of most computer files is haphazard. For some ephemeral files that we are prepared to lose, this could be acceptable. For files that represent data that has long-term importance, it is unsatisfactory. Well-managed creation, storage, maintenance, archiving and destruction of files requires a conscious strategy and sound operating procedures. A common-sense approach must be brought to the handling of all data within an organization.

Every user of computers has file management responsibilities, as well as a major vote in deciding how this job will be performed. We believe that those who are hired to operate computer systems should advise and assist users, not impose a particular workflow. If users do not buy into a consensus of operational procedures, there will be failures. This tutorial is intended for those users and managers who are willing to work together to maintain collections of files which support their activities. It is also relevant to individuals who manage their own systems. The ideas and software tools discussed will not eliminate the work of file management but will, we believe, make that work more effective and less frustrating.

It is appropriate to note what this tutorial is NOT about.

First, we will not be delving into databases. Database files are just one more set to be managed.

Second, while we will talk a great deal about methods and tools, we will not try to be encyclopedic in listing particular products or pieces of software. New software appears daily.

Third, while we will consider both strategies and tactics for manipulating a collection of files into an acceptable order and for maintaining that state, we will not discuss in detail all the possible filing rules and conventions. File systems should serve their owners, not vice-versa.

1: Preliminaries

1.1 Files

Computer files are the basic objects of interest in this tutorial, and we need to define carefully what they are for the purposes of discussion.

Computer file: an identified block or set of blocks of data storage capacity on some data storage medium.

Note that our definition allows that a file may or may not contain data. It is simply the mechanism or object used by the operating system(s) of our computer(s) to identify a collection of words (bytes) of data storage so that this storage space can be accessed when necessary.

Wikipedia gives a similar definition (https://en.wikipedia.org/wiki/Computer_file):

> A computer file is a computer resource for recording data in a computer storage device, primarily identified by its file name.

The two definitions are very similar. Neither talks about the **usage** of the objects we call *computer files*. Yet it is the usage of these objects that is the reason they exist.

Examples of the usage of computer files:

- program source code in some programming language such as FORTRAN, PASCAL, C, Java, Python, R, BASIC, or many others
- database management commands such as those written in SQL (Structured Query Language) that drives software such as MySQL or PostgresQL or Oracle DBMS
- database content, which may be stored in many forms
- Textual documents, which may be stored in many ways. These may or may not specify the formatting of the textual document.
- mixed media documents, such as those representing a book that includes graphics
- photographs and other images
- audio recordings
- video recordings
- drawings
- instructions to machine materials or 3D print objects
- streams of machine instructions that computers can execute. These are typically called **programs**

Filing cabinets serve as an analogy to computer file storage systems but that analogy is imperfect. In a filing cabinet, a file is much like a single item such as a picture, letter, memo, or report stored in the filing cabinet. Most filing cabinets use manilla folders to hold the documents which are stored. What goes in the folder can be pictures, letters, recipes, half-eaten sandwiches or whatever. Filing cabinets serve to store many manilla folders. Some folders may be empty. In some cases the things we want to place in them do not fit properly. For this reason, there are different sizes of folders, drawers and cabinets. In computers, we have different storage devices and the differences may be very important to the efficiency with which we can store and use our files.

The essential difference between the filing cabinet and a computer is that a computer does not just store different blocks of data, it can act upon the content of the files, transforming them or even altering the file itself. The scope of such actions is continually evolving.

Filing cabinets also serve to organize the folders. The main purpose of this organization is presumably to allow for ease of retrieval of wanted folders. This will also be one of our main goals in regard to computer files.

1.2 Directories

In order to organize collections of objects (people, books, auto parts, computer files), it is usual to define a hierarchy of headings under which items are classified. We will use the term DIRECTORY to refer to the list of items at each level in the classification. In particular, we will use the following definition:

Directory: a file which lists and points to, hence "contains", other files.

In the Apple Macintosh, such a construction is called a "folder", with files being like the individual "letters" inside. We have already mentioned the manilla folder which will serve as an analogy for a directory. This can be extended so that a filing cabinet drawer or a partition within a drawer is considered an even higher level of organization, with a list of the contents of the drawer or partition at the front showing how to find the individual files.

There are a number of different conventions used by different software developers for names of files and directories. WordPerfect, Word, and some other systems use the term "document" for a file. However, they may not include in their consideration the program files used to handle such "documents". In our treatment, we must take a global view, allowing us to consider the management of all data storage capacity on a collection of computers.

As an aside, it is worth mentioning that most Word document files are actually specialized computer programs that "run" when we try to read them. The program code is used for various reasons such as adjusting the display. However, the fact they are programs has led to hackers embedding malware inside what should be simply a body of text and formatting information.

We will consider **directories** to be files in themselves. This view is consistent with the way in which directories are used in computer systems. After all, it takes space to write down (store) lists of files. In the filing cabinet analogy, the list can be placed in a manilla folder at the start of the drawer or partition.

Furthermore, we can think of directories (lists of manilla folders) pointing to other directories. This defines a TREE structure for the entire set of files. The directories within directories or sub-directories, eventually contain only files which contain data (or empty storage), though we may have empty directories. Such empty directories may be set up in anticipation of some files being created.

On most computer systems that store data (as opposed to computers that control machinery, communications, etc.), it is common to consider the "starting" point of the files. This is generally referred to as the ROOT directory which contains all others.. Individual users generally have a "main" directory somewhere in the structure. On Unix systems (Unix, Linux, BSD), it is common to place these under the directory **/home**, so that "john" will have a HOME DIRECTORY **/home/john**.

Within the overall tree structure, there may be several copies of the same file or several files with the same name, each in a different directory. We will NOT allow the possibility of duplicate names within a single directory, as this leads to confusion and we know of no computer operating system which will allow such duplication. To identify just one of the files from the set having the same name, for example MYFILE, we need to give the set of directories which define where it is stored. Starting at the root directory, we can list the directories as we descend the tree structure in order. This PATH from the root directory to the file of interest (MYFILE) is unique for each copy of the file, for example:

root –> LETTERS –> BUSINESS –> Y1989 (contains MYFILE).

The path in many systems is usually specified for the present example as

We write this path for the present example as

/LETTERS/BUSINESS/Y1989

The complete FILE SPECIFICATION, or filespec as we shall abbreviate it,

The complete **file specification**, abbreviated **filespec**, also called the **fully qualified filename** is

/LETTERS/BUSINESS/Y1989/MYFILE

This uses the convention that the forward slash (/) is the DELIMITER between directory names. However, Microsoft operating systems (MSDOS, Windows) generally use the backslash (\) as the delimiter. This can be a source of annoying trouble, though the issue is mostly dealt with by modern software.

1.3 Volumes

The hierarchical (or tree) structure is an important concept for most computer operating systems. Moreover, it can be further applied to each VOLUME of physical data storage.

> **Volume**: a single logical storage area for accessing data. A volume will have particular properties concerning its capacity, speed, and limitations on file size or file names, or other restrictions.

The filing cabinet itself serves as an analogy for a volume. Alternatively, we may think of a book made up of chapters (directories) and the chapters made up of pages (files). For the present work, examples of volumes are a DVD, a USB flash drive or partitions of a physical disk or a solid state storage device. Because of the variety of mechanisms and devices for data storage, "volume" may sometimes refer to the physical object rather than just the logical storage area, which may be a partition of that object. Operating software may also hide some of the details in the interface to the data.

1.4 The size of files

In our filing cabinet analogy, there are clearly physical limitations on the size of the objects to be stored. In filing cabinets, almost all the items stored will be of a similar size. However, in a computer file system, files may be as small as a few characters to as large as many billions of characters. Moreover, the idea of a document as a physically unified object in a paper file system does not always apply. When working on a report, we may CHOOSE to structure it as a single object or as a set of separate pages. This raises questions of strategy: a book manuscript can be stored as one file, in chapters or even individual pages. Indeed, at different stages of writing, the user may want to work with all three structures.

Limitations on the size of computer files are not only imposed by the number of bytes the devices can store but also on how the devices access those bytes. Each device has to translate file names into addresses of blocks of data in the storage space and the ADDRESS SIZE and BLOCK SIZE therefore restrict the size of files. A street with 3 digit addresses can have a maximum of 1000 houses. If each house is allowed a maximum of 5 people, then we can house at most 5000 in the street. Similar calculations apply to computer storage. See https://en.wikipedia.org/wiki/File_size

When file size maxima are exceeded, special measures must be taken. Modern filesystems have largely eliminated such concerns but occasionally work with older files raises an awkward reminder. Most users will only encounter size issues when copying a large file to a set of archival media e.g., DVDs. In such cases, we recommend finding suitable backup software that is set up to carry out this task or someone who really understands what they are doing.

1.5 Some notes on how computers work

At the heart of every computer is a processor. Connected to the processor is a bank of random access memory (RAM), to and from which the processor can transfer information. Some of this information can be treated as instructions for the processor, which cause it to perform various functions. To get the computer going, we have to have some instructions already in the machine when it is first switched on. These are stored in Read Only Memory (ROM), though in most modern machines "ROM" means "read only until we reprogram it".

Also connected to the processor are input/output channels or ports. These allow the computer to communicate with the outside world. The "random" in "random access memory" does not imply haphazard. The memory is, in fact, generally organized in a highly structured way so that the processor can find any part of it quickly and select pieces of information from any part of the memory as quickly as from any other. Information stored on a tape is not "random access" since we must wind the tape along to the relevant data. A blackboard is a better analogy to RAM; even more suitable is a blackboard on which squares have been ruled.

BUFFERING is the process of collecting data in memory into a suitable chunk then transmitting this into or out of the computer. This happens because different devices require certain sized chunks of data to be brought in or out, or because transmission lines are "busy", or because RAM is fast and devices are slow. In the last case we save up output and transmit it all at once. Whatever the reason, we need to make sure we FLUSH buffers to ensure data is saved where we want it. Commands like **flush**, **unmount**, **eject** or **safely_remove** are used to do this. How this is done varies with the operating system and its interface. A typical operation is to "Right Click" on the icon representing our external device (actually likely a partition) to display options which should include unmounting. Unfortunately, it is very easy for users to simply pull USB flash drives out of their sockets. This almost always causes damage to data, if not to the devices themselves.

1.6 File permissions and special types

Most operating systems record extra information about files. They may, for example, flag whether a file is to be protected from modification. A **read-only** flag is very common but users should note that it is easily changed or bypassed.

Unix systems keep track of the **owner** and the **group** to which a file belongs, as well as noting whether it is **executable** (that is, a program that can be run by the computer). There are flags that control whether the owner, group, or "others" can read, write, or execute a file. We would suggest that permission settings are the most common cause of users experiencing glitches on Linux systems, though most such issues are quickly resolved.

Generally changing flags for these controls on files needs administrator or **superuser** status. Some novices try to run everything as the administrator but this can be dangerous as it opens the door to hackers and malware.

Operating systems generally do not remove data from storage devices when users "delete" files. Instead, the filename is changed e.g. "myfile" becomes "~yfile", or the filename is moved under a "Trash" or "Deleted" directory. Thus the data is NOT erased, making it readable by those who should not have access. On the other hand, we can "undelete" or "restore" deleted files.

If we really do want our data erased, we must "Empty trash" or in some way purge the data. Even so, most storage devices may leave physical information that can be recovered with special technology. Special tools exist that aim to make such recovery impossible, and we discuss some briefly later.

Most operating systems also allow "hidden" files. Truthfully, these are just files with some particular naming structure, commonly a "." as the starting character of the filename.

There are reasons why software designers want to hide files, mainly because they are not really needed by the users. However, if they are important to our operations, then they should be visible so we can back them up. We can, for example, copy .myfile to dot_myfile if we want a visible version of the file but need to be aware that changes to the original hidden file (possibly a program without user intervention) are not made to the visible version.

We should also make note of symbolic links – files that are pointers to other files. Generally, acting on a symbolic link file "slmyfile" that points to a different directory or even volume with file "myfile" will behave just as if "myfile" were being used. There are dangers in this, since we may not realize "slmyfile" and "myfile" are the same and damage some information we want to keep.

1.7 Timestamps and other metadata

All common operating systems apply one or more timestamps to files. Generally, we will want the time a file is last modified, so that the version of a file with the latest timestamp is then the most up-to-date.

From Wikipedia:

> Practically all computer file systems store one or more timestamps in the per-file metadata. In particular, most modern operating systems support the POSIX stat (system call), so each file has

three timestamps associated with it: time of last access (atime: ls -lu), time of last modification (mtime: ls -l), and time of last status change (ctime: ls -lc).

Clearly, if files are being moved **between** different computers, it is going to be important that the clocks of those computers are synchronized, and this is a non-trivial task. Fortunately, there is Network Time Protocol (https://en.wikipedia.org/wiki/Network_Time_Protocol).

> NTP can usually maintain time to within tens of milliseconds over the public Internet, and can achieve better than one millisecond accuracy in local area networks under ideal conditions.

How the timestamp is stored also affects how well we can determine which file is most recent, since the length of the field storing the timestamp will limit the resolution. For example, the EXT3 filesystems that were common for Linux had a 1 second resolution but the newer EXT4 has a millisecond resolution and NTFS claims 100 nanoseconds. However, the internal mechanisms by which timestamps are applied take time to execute, and we would caution against trusting timestamps closer than a few seconds.

2. Planning and executing file management

In this section we suggest you "look before you leap" in file management.

2.1 Design your directory structure

The structure of the directory tree for a particular ensemble of related files can affect how much work we must do to operate on the files. Too many levels (i.e., subdirectories of subdirectories) and too many divisions of a given directory can lead to a great deal of changing directory (`chdir` or `cd` commands or clicking on directory names in a file manager). On the other hand, too many files in a single directory means we need to search or scroll to find the one we want.

2.2 File naming conventions

We do not trust timestamps to reflect the true time at which they were last substantively modified. Users can easily add a space or carriage return or inadvertently "save" in some way. Therefore it is good practice, if the modification date or creation date is important, to put the date in the name. For things like bill payments, we like to put the date at the start of the filename, e.g., 20230228PayVisa.pdf

2.3 Deciding a backup strategy

Computers make it far too easy to delete data, or to otherwise make it inaccessible. On one occasion the telephone rang just as a "chown" command (change owner) was being issued to allow access by one of us to files that were created by another. The distraction of the telephone was such that the "chown" was applied to the system software. This meant no program would run. In this case no data was actually lost but a lot of work was required to save it to backup and reinstall the operating system. Do avoid this catastrophe if you can.

Both Windows and Macintosh have continuous backup options. These supposedly act more or less on the fly and copy our information to remote servers. See below in "Network Data Storage Considerations". There are costs in local computer activity as well as network usage, and those of us with large amounts of data are going to have to pay for storage as well. And we may need to wait a very long time for large files to be copied over even fast network connections. Linux has, at the time of writing, no fully developed option such as the Macintosh Time Machine. However, there are a number of tools and even customisable scripts to make frequent incremental backups.

Two major objections to automatic continuous backup are:

- there may be legal restrictions on allowing copying of some information, for example, medical files in a physician's office;

- many (most?) of our files are not worth backup.

Thus it is worth considering WHAT we choose to back up, and putting that information in particular directories that are flagged for backup. Backup software asks that we specify what directories to back up, and usually also lets us exclude files of a particular type.

A difficult issue is that we should back up our computer configuration, that is, the operating software, utilities, programs we use, and – most of all – our settings and preferences. There are tools to do this but the overlap between the "system" material and our "personal and professional data" is sometimes muddy. In Linux systems, the **Timeshift** program is intended for this purpose; while it is easy to set up and run, we believe truly understanding what it is doing requires some effort. On Windows, the **System Restore** utility has a similar purpose. Note that both these tools require that the appropriate setup of restore points is needed. Apparently (we haven't had a chance to try) **Time Machine** on Macintosh will also do this.

Apart from system files, users generally have a lot of files, and our estimate is that the majority of these are ephemeral.

Backup tools

Clearly backup is important. Apple makes it almost a default, to the extent that the auto-backup has been used to recover data on criminal activities when the local files were later encrypted. Some Linux distributions (e.g., Linux Mint) now set up a TimeShift backup for **system** files as a default. Data files like the home directory have to be explicitly included. We find this facility uses a lot of storage space, sometimes failing because it exhausts our device space. However, we tend to have a lot of "experimental" files such as virtual machines that use a lot of storage but which we do not value enough to want to back up.

Windows now includes a backup and restore facility.

Linux offers a number of backup tools but we tend to use **rsync** https://linuxize.com/post/how-to-use-rsync-for-local-and-remote-data-transfer-and-synchronization/ We have integrated this into the Double Commander two-pane file manager with a set of custom icons (we can choose rsync copy, copy only visible files, copy all the left hand tree with deletion of extras that are in the right hand tree). We can do this with the right hand tree being an external drive or even a network location.

There are many tools for backup, and in particular work environments it is worth considering carefully which to use. Once again, we will urge that the backups be checked and that the software be tested. In our experience, specialized devices have been less than perfect, and the learning cost of particular programs is high enough that we prefer the simpler rsync copy.

Some suggestions

- Version controlled files (see below) using external and reliable platforms such as Github or Gitlab have *de facto* backup in the version repository, so only need saving locally to have a snapshot of a project.
- It is likely worth putting all directories and files to be backed-up under an umbrella directory e.g., "MyFiles2Save/". Our work constantly changes, and the work of editing the list of directories to back up becomes an onerous chore. On the other hand, we need to be aggressive in only putting wanted material in the directories under our umbrella.
- It is worth using a "temp" or "working" directories that are NOT backed up for experiments and trials or tasks that we know are not worth keeping. It is easy to copy the few files we DO decide to keep to a directory that we regularly back up.
- We find it helpful to locally work with certain files, clean up the relevant directories, then copy to a (local) network attached drive. See below regarding NAS. This strategy gives us a chance to mess up the local files but recover them from the NAS. We also back up the NAS files periodically to external media.

- We find it easiest to do backup using the Double Commander file manager with a customized **rsync** ("copy more recent") tool. However, this is NOT automated nor periodic. It suits our working methods but others will find it risks failing to back up important material.

2.4 Selecting a tool set

Do NOT listen to people who say "You ought to use . . . ". It is YOUR computer. We believe technical advice on what software is helpful should be phrased "You may want to look at . . . .". Certainly we have our own views, and some of them have been voiced here. Some general considerations:

- You will want at least one file manager. We tend to use Double Commander, but that choice is driven by the "extra" tools we are prepared to spend time adding. We also occasionally use the default **Caja** and sometimes Midnight Commander for specific special details we will not describe here.
- While most users will NOT work with the terminal (command-line) interface, it is recommended that you know how to get it running. In the event you need to fix some glitch with the help of a support person, particularly by telephone, they are likely to want you to use this so you can issue diagnostic or repair commands.
- Unless you use your computer purely for web-surfing, you will want some sort of data backup. System backup is also useful but we generally do not bother with it since we have multiple computers. We have been known to boot a damaged machine with a live-USB (operating system on a flash drive), copy off data files to an external disk and reinstall the operating system. But we do have operational machines to use in the meantime.
- At some point, you will want to choose tools for things like deduplication, file compression, etc. Don't be afraid to try out possibilities on an experimental set of files/directories to see how they work, then choose what you like. Conversations with other users may be useful to all of you, especially if you have information from your own trials.

2.5 How does the user issue (file) commands

There are a number of ways in which a user can issue a command that acts on a file

- typing instructions in a command-line terminal
- clicking or double clicking on a file in a file manager window
- clicking and dragging a file icon to an icon of a directory or other tool
- choosing a program from a menu and answering questions in a dialog, or opening the file under a "File" menu (often at the top-left of the program window)
- highlighting a file in a file manager and pressing the "Enter" key (or in some cases other keys, like "Delete")

3. Workflow and working style

Let us consider different ways of working with computer files.

3.1 Command line vs. GUI

Command line interface (CLI)

An early mechanism for issuing commands was a keyboard and display, either paper or a screen. Indeed, modified typewriters and teleprinters were much used up to the late 1970s and these **terminals** gave rise to the concept of a "virtual terminal" as a rectangular area on a screen upon which text is displayed and commands are typed. We still use "terminal" as the name for programs that do this, though the description **command line interface** (CLI) is also common. In Microsoft Windows, the program CMD.EXE is such a terminal. In Linux, the machine upon which this paragraph is being entered offers several programs that provide a terminal; the default is called `mate-terminal`.

When a terminal has been opened, operating system file management commands can be typed. Generally there will be **built-in** commands that are part of the operating system but the difference between these and programs that are **executable file programs** is only of importance if those files are somehow corrupted so that they behave in unanticipated ways.

These commands have a particular **syntax**. In Linux

```
cp  afile.typ afolder/
```

will copy the existing file `afile.typ` in the current directory in focus to the existing directory `afolder` relative to that current directory in focus. The command `cp` also has a number of options that alter its behaviour. Sadly, the names of commands are not the same in different operating systems. In Microsoft DOS/Windows, the copy command is `copy` and its options are different from those of Linux `cp`.

The most important use of CLI programs is within scripts – sets of commands we execute automatically. Using individual commands for regular file management requires a lot of typing, with attendant risk of errors.

Typical commands

Here we will use the Linux (actually **bash** or Bourne Again Shell) commands to illustrate the typical capabilities of the command line interface.

- creating files: `touch` (which can also update the timestamp of an existing file)
- copying and moving files: `cp` and `mv`. The `mv` can also be used to rename files
- deleting files: `rm` in Linux, `del` in Windows. In some older operating systems there were many choices such as `ERASE`, `KILL`, `DESTROY` and even `UNSAVE`.
- learn the properties of files: `stat`
- change metadata: `chown` changes the owner of a file; `chmod` changes permissions. Note that there are other commands.
- listing the contents of directories: `ls` in Linux, `dir` in Windows (note that `dir` can also be used to search for files by name)
- displaying the content of text files: `cat` in Linux; `type` and `copy` serve similar functions in Windows.

Programs such as word processors or image display programs are launched by name, with filename(s) as parameters. There may also be options. One of the nuisances of the CLI is that the user must learn the effects of different options.

The tools for copying, moving, renaming and deleting files are just programs that are generally part of the operating system. Note that a move is a copy followed by a delete of the source. Similarly, moving a file to one with a different name renames it.

Sometimes it is useful to employ external (non-operating system) programs to perform the functions normally carried out by the OS commands. An example is the `rsync` program which primarily copies files that are more recent or do not exist in a target directory. It has a great many options and we do not recommend its command line use but prefer to use it in scripts. See https://en.wikipedia.org/wiki/Rsync

Graphical file managers

Most computer users access their files via a graphical interface. This is actually a named program but often users will perceive it as "the operating system" because it is so central to their work with a computer.

In Windows systems, the default file manager is **File Explorer**. In Apple Mac systems it is the **Finder**. In Linux systems, there are a variety of file managers, with the default depending on the distribution, though users can (and do) frequently change this. Common choices are **Nautilus**, **Thunar**, **Caja**, **Konqueror**, and **PCMan** but there are many others.

Typically we see files as icons on our screen and we move them by clicking and dragging them, or else by highlighting them and using keyboard shortcuts. Generally we can choose the layout of file icon displays. We happen to like "List" mode, where the name, modification date and size are reported and we can organize the list by size, name, date etc.

The importance of keyboard shortcuts

Clicking and dragging is quite inefficient for serious file management operations. The keyboard shortcuts streamline the tasks. Possibly the most important three of these are

- `Ctrl-C` (Hold the `ctrl` key and press the `C` key) to copy a file to the copy-paste buffer
- `Ctrl-X` to move a file to the copy-paste buffer
- `Ctrl-V` to paste the file (contents) of the copy-paste buffer to the location or directory in focus.

Each user must decide for themselves which shortcuts to learn and use. By and large, I find that it is worth knowing perhaps a dozen that are used frequently. Because different programs use different shortcuts and these often interfere with each other, I find it not worth the effort to learn too many.

Two-pane file managers

Many computer users (including the author) favour the use of dual-pane file managers. Multi-pane programs also are available but add complexity to workflows. The advantage of dual-pane file managers is that files can be marked (selected) in one pane and a command issued that acts on all of them. This is particularly helpful for copying and moving files, for comparing them, for processing a set of files.

Dual-pane file managers have been around quite a while and an early version was the Norton Commander, which used a text-based interface, either on the full screen or in a terminal window. Such text-based interfaces persist, for example, in the **Midnight Commander** program available primarily for Linux, though there are binaries for Macintosh and Windows. In Linux, this program is abbreviated `mc`. It works well but the text display is somewhat unattractive and possibly tiring on the eyes.

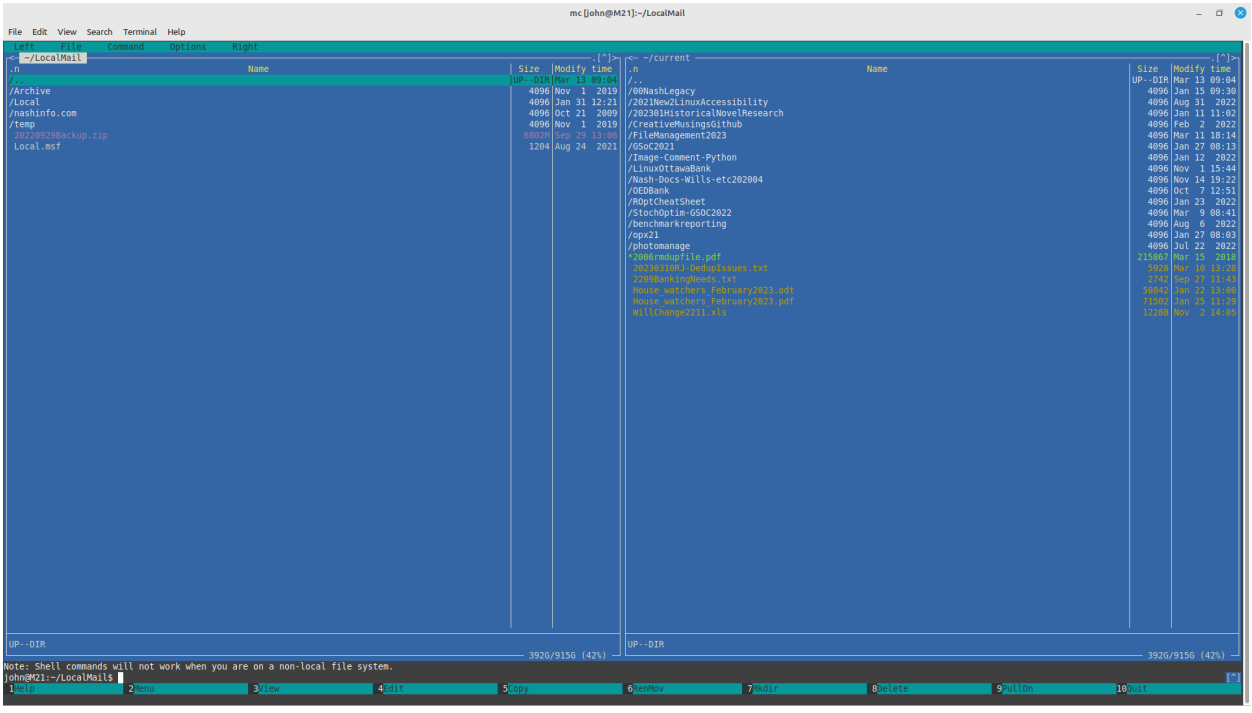

Figure 1: Midnight Commander (mc) file manager

More modern dual-pane file managers exist using a graphical interface. One that is available for Linux, Windows, Mac and BSD is **Double Commander**. This also allows users to add icon driven programs that can operate on highlighted files or directories.

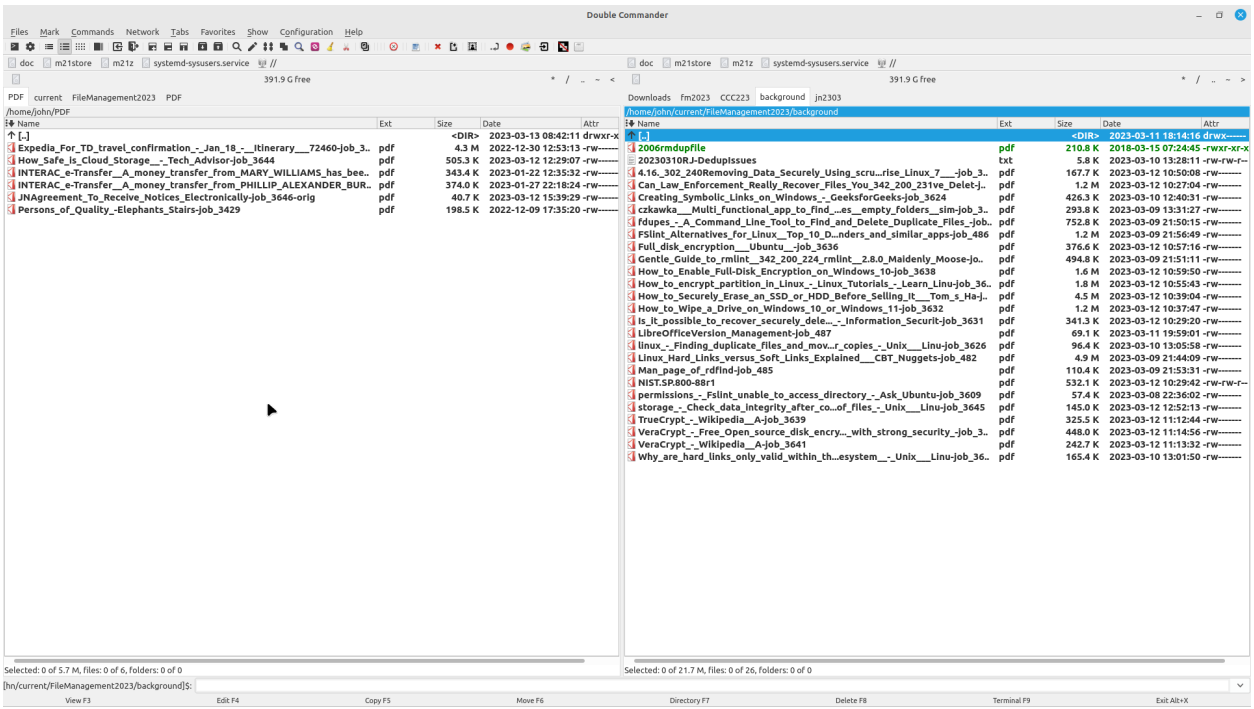

Figure 2: Double Commander file manager. Note small custom icons at the top.

Workflow standard practices

When performing a lot of file operations, it is very easy to make errors that result in data loss. Developing standard practices can reduce the probability of such errors. Some standard operating practices that can be considered are:

- copying or moving files (in a dual-pane manager) is always from the left hand side to the right. This is only enforced by user habit. We can attest that breaking the rule can break the files!
- It can be helpful to preface filenames with a date, since file dating is often less than reliable. If the date is in the form YYYYMMDD, then the files will sort naturally into increasing date order.
- copying should mostly use options that only replace files if the replacement is later, or else the replacement is manually approved.

This last requirement is most easily accomplished by using a special utility called **rsync** (https://en.wikipedia.org/wiki/Rsync). Sadly, the versions for other than Linux are less than perfect, though possibly usable. In my own work, I have an icon added to Double Commander to apply `rsync` to copy the active left window folder to that on the right, with a second that only copies **visible** files (those that have names NOT starting with a `.`).

3.2 Scripts – for those willing to write them

While there are some power-users who prefer to work with the command line interface using more or less the operating system commands, we are of the opinion that these commands are best used within scripts that can set the options, perform checks on inputs and provide information to the user in the form of messages or log files.

Most users do NOT want to write scripts. However, we strongly recommend that users do learn how to make a script from single or small sets of commands with specified options. They are simple text files. Generally we give them “executable” permissions and store them in special directories for executable programs.

Example of script writing

The type of script general users are most likely to prepare will be those that are relatively simple but require unique syntax that is tedious and error prone to type directly.

Here is an example of a script to mount the disk of another computer (called M21) that is on our local network. The machine address, user, and the particular directories to be used for the remote volume and local mount point must be provided precisely. We also want to check that the mount operation is successful i.e., our remote machine is turned on and connected to the local network.

```
#!/bin/bash
# above line says this is a script to be executed in the bash terminal
# c-M21 -- the name of the script
 mhost="M21.local" # remote machine address which is translated by network
sshfs john@$mhost:/ /home/smnt/M21 # the sshfs connection/mount command
echo "" # display a blank line
echo "Check it is mounted" # a title for the check
mount | grep M21 # mount command lists mounted volumes, grep searches for M21
read -p "OK?" TMPSTR  # waits until user hits Enter
```

3.3 Finding information and fixes

Nobody knows every command and option. In fact, even experienced users must continually check syntax and options and learning how to do this efficiently is an important skill.

Within an operating system there are ways to find out about commands. In Linux, for example, on can generally type a command name followed by a space and the option `--help`. e.g.,

```
cp --help
```

Also one can use > `man cp`

Interestingly, `help` shows many commands, but `help cp` says there is no help information about `cp`.

Windows has similar commands for the command shell (program CMD.EXE). A search with Google or similar search engine using "getting help with windows commands" will provide information on how to get the syntax and options.

Internet search is, of course, important for learning what commands are available and their syntax and options. It is worth practicing searches to gain some skill in finding the precise information we want.

Sometimes documentation or search results are not enough to allow us to give the correct command. In such cases, various forums may be helpful. For example, Linux Mint has the excellent https://forums.linuxmint.com/. The more general https://stackexchange.com/ has a wide following and well-considered answers to queries.

Using such services carries with it an obligation to ask well-posed, courteous questions, providing output from the attempts we have made already and answering respondents as best we can. It is also worth mentioning that bugs and deficiencies are common in all computer programs and even if the program is correct, its documentation may have errors or omissions. Users frequently assume they are at fault when something does not work as expected. It is worth being politely persistent in seeking a good answer to our questions. This will help to improve the programs and their documentation.

We have seen how powerful the forums for open source software can be. On one occasion, I was trying to program a particular command to automatically locate a particular string in a file. After an hour of struggle, I posted a question with a shortened version of my code that could be tried. Then I went to get a cup of coffee. As the kettle finished boiling, I heard the ping of my computer. In less than 10 minutes I had a reply and within an hour a few more. It turned out the documentation could be read in two ways and my interpretation was the inappropriate one. By contrast, it is rumoured that bug reports to Microsoft take six weeks to reach a programmer.

User groups, particularly if they have mailing lists, can be helpful. Sometimes we make the acquaintance of those who can provide information, though we must be willing to learn, not simply ride on the coat-tails of others. And we must pay it forward and help others where we can. Ordinary users are often surprised that they are more familiar with certain programs or aspects of them than the IT professionals. None of us can know more than a fraction of what there is to know about a given system.

3.4 Personal documentation files

It is difficult to remember command syntax or the settings of graphically interfaced programs. Thus it is important to have mechanisms to remind us of these.

- A directory of small text files of "how to" notes, with filenames suggesting the subject of the files, can be helpful. An alternative is to email the information to ourselves, then save it in a mail folder which could be called "HowTo".
- Most modern operating systems allow "print to PDF". Saving appropriate information from internet searches or mailing list responses by "printing" them can be a tool to having information at hand. These files can be saved in the "howto" directory but it is important to rename them so their content is obvious.
- log or diary files can also be helpful but keeping too much information can make finding what we need time consuming. Such files can, of course, be put in a directory such as "2022Dec-logs-of-tries-with-program-XYZ" which can be reviewed and edited or discarded later but is out of the way of other files.
- Sometimes catalogues of files can help us to find where we have put data. This is especially true for catalogues of removable media. In the past, diskette catalogues were important, as these were our main data storage in the early days of personal computers. Nashcat was our own diskette catalogue program. Today there appear to be few such programs, though **Gwhere** for Linux (https://github.com/wdlkmpx/gwhere) seems to be an active project. https://www.zabkat.com/deskrule/catalog-offline-search.htm suggests there is a proprietary program for Windows.

Find and search

Whatever collection of helpful information we keep, we are going to need to have ways to find files with particular character strings in their names or their content.

In Windows we can use the Search box in Windows Explorer (File Explorer) to search either for strings in filenames, or by using "Advanced Search" the content. The Finder in MacOS allows similar functionality. In Linux, different file manager programs have different search capabilities. There is also, for example, the separate `mate-search-tool` that allows for both filename and content search.

Of course, "content" has to be textual for search on strings to work. If our information is images or sounds, a text search will not work and sometimes text is converted to images by programs. PostScript and PDF files can contain text either in text form or images depending on how the files are built.

4. Deleting and securing files

4.1 Simple deletion

As we have indicated, the deletion commands such as `rm` in Linux or `del` in Windows, or highlighting a file and pressing the "Delete" key (or Right-click and choose "Delete") all simply remove the filename from where it is located and move it to the "trash" folder. The Macintosh Finder needs the two-key combination "Command+Delete" to do this. Also note that "Right-click" does not exist on Mac, which has a 1-button mouse. Instead use "Control-click" i.e, hold the Control key when clicking.

4.2 Purging files

In Windows and several of the Linux file managers, highlighting a file and pressing "Shift+Delete" (hold Shift and then press Delete) will offer to permanently delete a file, rather than move it to the "trash" folder.

"Empty Trash", an option in most file managers, or sometimes via Right-click on the "trash" folder (it may have a slightly different name), will purge files from the trash (permanently delete them).

4.3 Encryption of files or volumes

We also may wish to secure files or volumes by encrypting them. This is a complex topic and there are frequent debates on the level of security of a particular encryption approach.

First, we need to decide if we want to encrypt a full partition (or a full drive) versus encrypting individual files. The tools are very different.

Second, while many users are worried about their information and choose to encrypt their "whole machine" (essentially their main storage partition), there are costs. The overhead of encryption and decryption is non-trivial, though many modern machines have specialized hardware included to assist. The human cost is likely more important, especially if we lose the encryption key.

Windows, Linux and Macintosh all offer partition encryption, typically on initial install/configuration. Windows BitLocker, unfortunately, apparently requires the user to log in with their Microsoft account, then saves the encryption key to a Microsoft server "in case you lose it". A great help to the FBI, MI5, CIA, FSB, etc. Users might be interested in **VeraCrypt** (https://veracrypt.fr/en/Home.html) which has versions for the major operating systems. On Macintosh, the volume encryption feature is called File Vault.

Directories (folders) can also be encrypted. In Linux **ecryptfs** https://ostechnix.com/how-to-encrypt-directories-with-ecryptfs-in-linux/ The encrypted information is stored in a file that is accessed as if it is a mounted volume. When the volume is unmounted, the information is not accessible.

We prefer to encrypt individual files and use the **ccrypt** program which is available on most platforms. Our choice is favoured in that we know the auther, Peter Seligman. https://en.wikipedia.org/wiki/Ccrypt Unfortunately, ccrypt is a command-line utility but we use it mostly via scripts, so we don't do much typing except for the encryption key. A GUI interface, **qqcrypt** works quite well (on Linux) but is not in the software repositories. http://qccrypt.free.fr/ There are some other tools listed at https://ccrypt.sourceforge.net/#other

For moderate security, various "zip" compression tools offer ways to password-protect files. Windows (in some versions but apparently not the "Home" ones) offers encryption as an option on Right-click.

File encryption is useful to protect just those files we want to keep safe. However, we are more or less forced to be proactive on **which** files we want to encrypt. In our own case, these are lists of important personal information, including password records etc. Moreover, we have developed some scripts to get all the syntax correct and in some cases to automatically copy the encrypted files to remote storage.

4.4 High security erasure of information

We have noted that the **physical** information of erased files may still be recoverable with special tools and equipment. Magnetic disk storage (spinning hard disks) need special attention because residual magnetization may allow data recovery. See the NIST "Guidelines for Media Sanitization" https://csrc.nist.gov/publications/detail/sp/800-88/rev-1/final

Multiple pattern writes to the disk surface are generally considered adequate to prevent data recovery. One tool is **DBAN** (Darik's Boot And Nuke). This is actually a small Linux distribution that is booted and will write patterns to a selected drive. It takes considerable time and there are risks of choosing the wrong drive, so we recommend performing DBAN operations on a "spare" computer. We tend to use an old machine that is kept for experimenting.

Another tool is **scrub** (https://github.com/chaos/scrub). This is part of many Linux software repositories. It allows a file or a device to be erased.

A somewhat less fierce approach suggested for Windows 10 and 11 by PC Magazine (https://www.pcmag.com/how-to/how-to-wipe-your-hard-drive) is a re-installation of the operating system with a disk wipe. A Linux install with formatting would likely be equivalent. https://www.pcmag.com/how-to/how-to-wipe-your-hard-drive claims that the non-quick format options of Windows will achieve what DBAN and Scrub claim to do. However https://www.tomshardware.com/how-to/secure-erase-ssd-or-hard-drive reports that tests show the erasure is not complete. This last article provided some good pointers on full erasure of disks.

https://fossbytes.com/best-hard-drive-eraser-tools/ gives some recommendations for Windows and Mac tools for erasure.

Ultimately, for very sensitive information, it is likely easiest to physically destroy the storage device with fire or hammer. In one case, we took apart a hard drive and ran a magnet over the disk platter. The platter itself is rather attractive, for example, as a Christmas ornament.

4.5 Accessibility vs. Security

We can secure our laptop by throwing it in a blast furnace. Nobody – including us – can then access the data. It is secure but not accessible. That is the fundamental dilemma about file security. Efficient use demands easy access. Good security requires limited access. Therefore, it is worth

- deciding which files are important to secure
- choosing the level of security to use: USB flash in a locked drawer, password protection with a Zip utility, file, directory or volume encryption with a serious cryptographic utility
- planning and implementing a mechanism to ensure backup of encrypted files to ensure we have access ourselves. Note that a typo in a password can make files unavailable, so it is worth considering layered backups i.e., we can access an older version if this happens. (We can attest that it does!)
- Keeping documentation of the procedures. People get sick and die! It is sensible to plan ahead with "In the case of my incapacity" letters.

5. Improving data storage for efficiency

5.1 Compressed "archive" files

When we have many related files it is useful to group and structure them. We do this by moving files to appropriate subdirectories within an umbrella folder/directory. However, there are often files we do not need to look at regularly. If there are a lot of these and they are small, it is likely there will be wasted storage. Files are allocated a space in terms of a number of blocks but the data rarely fills the allocation.

A useful solution is to assemble the data into a single file. One long-standing solution was the Tape-ARchive or TAR file. We no longer use tapes very much but the TAR structure is still important and we only have a single file, reducing the waste. Moreover, it is common to **compress** the TAR archive, resuiting in a **tarball**. These have the filename extension "tar.gz" if compressed with the **gzip** algorithm. There are other compression tools, so we see "tar.bz2" files, etc.

More commonly, the **zip** compression and archiving algorithms produce **zip** archive files. There are many tools to do this and they are now often part of the file manager. Right-click generally offers the option to "pack" or "zip" a file or directory. Sometimes we need to explicitly ask that subdirectories be included, as well as sub-sub-directories etc. (recursion).

Note that sometimes the tools will delete the original after making a compressed archive. Otherwise (which we prefer for safety) we must do this manually.

5.2 Migrating to new storage locations

Storage devices fail. In particular, flash storage (USB thumb drives, SD chips) tend to fail suddenly without warning. For that reason, we prefer "spinning" disks for storing important files we want to keep. These fail

too, but they tend to do this piecemeal. If a disk read or write seems to be taking longer than expected, it may be retrying the operation, indicating imminent trouble.

The only real way to avoid data loss is to copy or migrate collections of data to new storage. Fortunately, this is relatively easy nowadays with external drives. We use our two-pane file manager, plug in the new device and set the two panes appropriately, then run the copy command (in many two-pane file managers, the F5 key does this).

The paranoid may wish to check that the copy is faithful. Some ideas are given in https://unix.stackexchange.com/questions/495506/check-data-integrity-after-copying-thousands-of-files

Moving files from one device to another may help in defragmenting storage. That is, if we edit a file many times, its physical storage will not be a single contiguous set of blocks on the device. This used to be a concern and there were defragmentation tools within operating systems or available from third parties. Largely this has disappeared as a concern, since manufacturers now have sophisticated software in the device controllers. It may still be sensible to migrate files to simplify data recovery if the device fails in some way, since data recovery software generally looks at content. Having files in contiguous blocks makes that task easier. Truthfully, we rarely bother with this now.

6. Version control

Some projects require that we have a history of our work. Version control is the term that applies.

6.1 Filename version control

A simple mechanism for version control is to save each new version of a file with an appropriate filename. Thus 20230311JNdocument.odt would be a LibreOffice document by "JN" on the date shown.

This method is simple and effective but quite tedious. Nevertheless, it can be recommended for modest-sized projects with participants who are willing to follow a simple but demanding discipline.

We note that we have ALWAYS come to grief when a group of workers has tried to email files to each other. Inevitably, someone edits the "wrong version" or there is a fight over which is the "right version". Do not even think of emailing files unless they are properly identified as to author and time/date.

6.2 Version control systems

For bigger projects, software writers have come up with some very sophisticated tools. These allow multiple workers to act on many files at the same time yet keep the overall collection of files in proper condition. Let us not pretend this is easy or without lots of work. Nevertheless, projects like the source code collection for the Linux kernel could not be run without such tools. In fact, Linus Torvalds, who gave us the Linux kernel (the proper name for the full operating system is GNU-Linux, since the GNU tools provide the interface) also gave us **git**.

git, **Mercurial**, **svn** and a host of other version control systems exist. We use git and svn in our own work.

In this tutorial, we will not detail how these tools work but will note that they are at the core of some very popular services such as **Github** and **Gitlab** which we use regularly for projects in which we are involved. And, yes, we regularly must work hard to resolve glitches caused when two versions have conflicts that must be resolved, a process that requires awkward manual intervention. On the other hand, we can relatively easily trace the progress of a project, with tags indicating who did what and when.

6.3 Version control within particular applications

For many users, more specific version control may be preferred. LibreOffice and Microsoft Word offer ways to keep track of changes to documents. We have not used these directly and believe that if they are to be used, it is worth making sure the procedures are in place to identify the workers etc.

Note that some programs offer version **history** but lack the merge and conflict resolution of true version control. Beware of false hopes.

7. Deduplication

We often have multiple copies of files in various locations. How do we remove the extras?

Characterizing the deduplication problem

There may be a number of features important to the problem at hand but a particularly useful characterization is whether the suspected duplicates are interspersed with the "master" files or in a separate directory/folder/volume. Unix-like systems (unix, linux, mac) treat the whole file set that is MOUNTED (i.e., accessible to the active user) as a tree, starting with the root "/". Branches (paths) are then considered as folders or directories. Consider two such branches A and B. A might be /tempdisk/mydepartment/2022documents/ while B might be /masterstore/mainfiles/mydepartment/alldocuments/. As long as there is no overlap between the branches, none of the files are interspersed.

Alternatively, we might be wanting to clean up branch B that has copies of files (possibly with different names) mixed up in many sub-directories.

Some other concerns:

- What is a duplicate? In my rmaiib.pl program (see below) we consider that two files are identical only if they have the same length in bytes and these bytes match exactly. This is what the *nix command "cmp" does. However, some other workers use a hash function of files and consider equivalence to be equality of the result of applying this hash. In practice, it is likely safe to use the hash function approach and it is probably a good deal faster.
- Image, video and audio files can have many different formats and compressions (usually specified by coder-decoder rules or codecs). Finding "duplicates" or files that are sufficiently similar requires care and attention and is generally quite difficult, though tools are becoming available.
- Symbolic links are file names that point to other file names. They don't contain data. In the case we wish to remove files from A that are in B, a symbolic link in B pointing back to A will cause us grief.
- Filesystem access can create some concerns. Unix allows more possibilities for the infrastructure of data storage, usually referred to as "filesystems". Windows generally uses the (now dated) FAT (FAT16, FAT32, exFAT) or NTFS filesystems but there is an alphabet soup of possibilities for *nix systems. The user needs to be able to mount the desired device and folders with appropriate privileges (e.g., to delete, copy or move files). In certain operating environments, this may present challenges, or at least a need for care. We recently overcame one issue with the (very nice) dedup program **czkawka** (Polish for hiccup) which would not access a ZFS format master file store due to sandboxing restrictions (security of access controls) in the program packaging. We overcame this with a different program packaging and also by compiling it from source code (the true geek solution).
- A trick to conserve space is to use "hard links" to have more than one filename for the same physical data in storage. These multiple pointers to the same information can be in different subdirectories in the tree. There are tools that will find duplicates and point all the filenames to the same actual data, allowing reuse of the space no longer addressed. Tools to do this need to be well-written so they can be trusted! You will have no reduction in the number of filenames and a bit of overhead to store them.

Tools

There are a great many tools for deduplication, though most of which are for the case where duplicates are interspersed in a single branch of a collection of files. This generally implies that the user has to intervene to decide which is the "master" so the others can be removed.

Interspersed duplicates

dupeguru (https://dupeguru.voltaicideas.net/) is a free and open source tools that works on the major platforms. We tend ourselves to use **czkawka** (https://qarmin.github.io/czkawka/) which is a successor to the now-deprecated **fslint**. It is also free, open-source and cross-platform. Note that we did encounter an issue with the **snap** installed version due to file a access glitch with the snap infrastructure.

Separate file trees

For the ReMove from A If In B (RMAIIB) case, we wrote a Perl language script prior to 2006. A significant part of the code tries to detect links from B to A and I believe this works correctly. However, if one is paranoid, it would be worthwhile checking all symlinks (the hard links should point only within the branch in which they are located) and possibly copying them to create regular files in a special directory to be further checked for duplication. There are some mailing list discussions of the RMAIIB problem, but we have not seen any other program for it, especially given the symlink concern. I can make rmaiib.pl available. It should run on Windows and Mac, but we have not tried to do so for over a decade. It needs a Perl interpreter, and these are available free but IT people sometimes object to installing. On Linux/Unix, Perl should still be "part of the furniture", though Python is now more popular.

In Figure 2 for Double Commander, the yellow "brush" icon at the top of the screen invokes `rmaiib.pl` using the left and right pane paths as A and B.

8. Data recovery

Data recovery is something you don't want to have to consider.

"Do you have a backup?" is the question all IT support staff will ask. The answer is usually "No" and the support person shrugs and suggests it is time for a coffee break.

If you do have a backup, you have a slightly annoying and tedious job of carrying out the restore operation.

It is worthwhile doing a "practice". We have known several cases where IT staff assiduously did the backups. When there was a device failure, it turned out the restore software or the backups saved were faulty. Given that restores are rare, it is not surprising that the software is less tested.

That's all we have to say. No sympathy if you didn't plan ahead.

9. Network data storage considerations

9.1 Cloud storage

It is popular to save data "in the cloud". Essentially this means storing files on a remote computer. If you care about people looking at your files, then this is not safe unless you control that remote computer or have all files strongly encrypted. See https://www.techadvisor.com/article/745187/how-safe-is-cloud-storage.html

Major computer companies offer cloud storage (Apple iCloud Drive, Google Drive, Microsoft OneDrive) and there are many offerings. **DropBox** was one of the early players and still active. **Sync** is a Canadian offering and has a free introductory account for up to 5 GB of data. That used to be a lot but we recently purchased a "64 GB" (actually 58 GB) USB-C flash drive for $6. Note that the flash drive locked in a fireproof box provides a solution we do not need knowledge of encryption to understand.

9.2 Network attached storage (NAS)

Storage on a network does not have to be remote. It is also useful to have storage devices on our local network, since it provides a very useful way to provide a common set of files to all users on the network, for example, all the people in our home or office. A number of manufacturers offer such devices but it is also quite straightforward to home-brew them with modest computers, either repurposed older ones or new single

board ones like the Arduino or Raspberry Pi, along with commodity storage devices. We have a Synology DiskStation.

NAS volumes behave more or less as local mounted volumes. Operations on files are limited by the network speed but for most applications the performance loss is almost unnoticed. The main operational concerns are setup and possible collisions between two users trying to write to the same file(s).

10. Archiving information

We have already mentioned the compressed archive file but in this section we will consider where to put files we want to keep but do not need at our fingertips.

10.1 Deciding what to keep and what to destroy

Most of us – especially the person in the mirror – keep too many files. On the other hand, it has saved some embarrassing losses from time to time. Nevertheless, there are files that are just going to be a nuisance in the long run.

Some questions that may help:

- Do we have any idea why we have certain files? If we cannot figure out what they are, then probably we should erase them unless they could be part of the current operating system.
- Do we need old versions of documents. Unless we need a history, they should be deleted.
- Has the ".cache" contents (typically browser history) been purged? These sometimes present a security risk since they record work we were doing.
- Should old projects be "zipped" and moved off-line? There are inexpensive USB interfaces for hard drives that allow old drives to be used to store "just in case" collections.

10.2 Where to store archives

As indicated, we can collect and compress directories into zip or similar archive files. If these are important, then it is worth storing them on multiple devices that are not in the same location. Fires and floods do not respect our data.

Cloud storage, preferably private, allows us to transmit the archives rather than physically deliver the storage media to a remote site. If physical media is used, then we also need to consider having a pool of devices so that we deliver one and bring another back to be updated. A disciplined procedure is needed for proper safety. Do we do this? Well, we try but to be honest we find it difficult too.

12. Epilogue

The follow-on from the suggestions above is in your hands. Computer file management, despite the many tools that try to make it easier, is a human activity.

References

Nash, J. C. Catalog Builder,Interface Age, vol. 7, no. 3, March 1982, pages 98,156-157

Nash, J. C. and Nash, M. M. Software and procedural tools for productivity enhancement in specialized data entry applications. Canadian Journal of Information Science, vol. 12, no. 1, 1987, p. 67-73.