noahs*ARK*~2009

Abstract Updated: **Z18#20091114SA~1200PM;** Portions of the following material are copyright by the respective owners.

o'SESAME~FileRoomClerk <> FileRoomCabinets

Supports Database Management & Organization of Shared Business Files..

The following digital filepath suggestions and restrictions have been tested and are stable under all four major operating systems.

Where do my files disappear to and why?

Are any of the following statements ever heard in your office? What happened to the 'Smith Contract'? Where is the '2009APR Financial Report'? Why can't I find the new 'HR-Termination Form'? Ever feel like someone has misplaced, stolen or deleted a file?
How much time do you and your associates spend every day looking for files, letters, emails, proposals, supporting data. Web links and more?
It is a sad fact, in business, education, government, military, home and institution - *the majority of desktop computer users are simply lost* when it comes to dealing with ***how and where*** files have gone that seem to repeatedly disappear into a black hole.

Typical Solutions

- Consequently, many application programs have been created for aftermarket sale, or included with the Windows, Unix, Mac or Dos Operating System (OS). Fast searches require huge index files that consume 5% to 20% or more of the total file storage space; plus the search engine grabs a chuck of base 'ram or dram' memory while actively indexing or searching.
- Meanwhile users are left to their own vices – randomly naming files with no understanding, coordination or relational consideration. *Why?* Because search engines typically find lost files with similar key or relational words, date range, size or type within the file name or contents.
- Users typically save files wherever the active program stores them by default; or where the user saved a recent file a minute or an hour ago. Consequently, the jajority of files are stored in a sub-folder of the application being used or under the Window's 'My Documents folder.
- Files that should be co-located with similar subject matters are randomly and haphazardly strewn over the hard drive by one or more users. inevitably, as computer users know to well, finding a file that was just saved five minutes ago can become a time-consuming chore.
- Relational subject associations are difficult and time consuming to index and relational key words are difficult to include within the name of every file; and are in fact, *blatantly and ridiculously redundant*.
- A historically better, but ***expensive solution*** has been Document Management Systems costing hundreds of dollars per concurrent user!

The File Room Clerk Approach

- None of the traditional search methods can match the speed of retrieval, functionality or security gained when a structured storage paradigm is combined with one or more search engines like 'Copernic', 'Everything' or 'Google Desktop (note: turn off web searching).
- Select the **[F9]** in the noahs*ARK* suite hot key at any time to review currently logged on user profile settings.
- *O'SESAME*~**FRC** stores and uses profiles for each user to provide powerfully linked application features.
- Applications and programs which use unique file extensions (jpg, pdf, doc, rtf, htm, etc.) may be configured for each user, independently of Operating System settings.
- Whether one uses *O'SESAME*~**FRC** or not, the following recommendations can improve the speed, reliability and integrity of file storage and secure access of very large business file libraries.

PAGES 1-5: *o'Sesame*~**FRC** is included and fully integrated within the **noahs*ARK*** portal.
For more information, be sure to visit **WWW.FCCSS.NET** or ContactUs Support@fccss.net or Call +1.770.395.6500

PAGES 6-11: **SUGGESTIONS AND REFERENCE NOTES ON THE MAJOR FILE-NAMING SYSTEMS.**
Materials are copyright or trademark by their respective owners.

Note: The **eveFRC** filepath features of *O'SESAME*~ **FRC** are compatible with Unix, DOS, Apple & MS-Windows operating systems.

Automated Features of *O'SESAME*~eveFRC

- The user is not required to find or create a folder structure
- The 4-level folder structure can be used in combination with filenames and sub-folders that are OS specific.
- By design, these first 4-folder levels may be used interchangeably by each operating system.
- Of course, users are at liberty to name a file at their discretion.
  Consequently, user-named files may not be compatible with every operating systems.
- The default settings for drive letters and master folders may be customized in the profile setup.
  See the **[F9]** *toggle* to set user profiles for the *primary* and *secondary* master folder paths.
- Refer to **[F1] [Help] [Mousepad]** screen for a complete list of all Hot-Keys. (See screen cap below)
  With nearly all operations, the user may modify, then paste a unique filepath when **[F8]** is selected - opening **FRC** directly on a unique **eveFRC** folder created for each Contact crm-record.
- **[Ctrl +F8]** opens Microsoft Explorer or an alternate manager on the Contact **FRC** folder.
- **[Shift+F8]** opens Search Engine on the Contact's **eveFRC** folder.
- **[Alt+F8]** runs a *macro* that auto-generates an **FRC** header, which may be included for documentation. Users select a file extension for the type of application required and a subject line if desired.
- By pasting the clip boarded default filepath into the [Save} box; the file will be automatically stored under the Contact **FRC** folder quickly.
- **[F3]** opens user's Word Processor on an expanded **Notes** file under the Contact **eveFRC** folder
  [Shift+F3] opens user's Word Processor within the Contact **eveFRC** folder
  [Alt+F3] opens user's Spreadsheet application within Contact **eveFRC** folder.

Each database table included in noahs*ARK* includes a non-repeating sequential record number.
The **SRN** is an indexed numeric field (cell) used to link, name, locate and manage related records and files.

There are two sets of four fields (cells) in each CRM-Contact database record (row) used to store values for Primary and Secondary FRC filepaths, which may duplicate one or more components.

- The first cell tracks the shared **FRC** hard drive letters, e.g. Primary: **N:** or Secondary: **C:**
- The second tracks the **FRC** folder name; **eve_FRC** is recommended and preferred.
- The third tracks the first sub-folder level **0-9 and A-Z**
- The fourth tracks the Contact's folder number that begins with a dash (-) plus seven of the SRN digits,
  e.g. **E\-0000557** **(Note: that's 9,999,999 Contact records - *expandable if needed*.)**
- The last field tracks a filename filter (default is ***.*** or ***.xls**, etc.)
- A fifth *virtual* field (cell) displays the complete value of the filepath. Example: **V:\eve_FRC \E\-0000557*.***
- When creating New Contact records, the Primary folderpath is automatically created.
  Selecting the **[~FileRoomClerk *menu*]** button on the Contact screen offers a list of user helpful options.
  Example: **[New or Update Contact folder]**
- While using any utility file manager that displays **FRC** folders and files, a user may simply note the dashSRN value, *Select* (click) on the **SRN field** located in the top left corner of a form-screen;
  Type the whole digits (i.e. 557) and click **[OK].** The associated Contact record is instantly displayed.

A Reference File is copied, automatically named and stored under each DFR numeric folder.

The filename is constructed from the 8-digit Contact record# plus up to 62-characters taken from the Common name field.
Acceptable characters are 'A to Z' and '0 to 9'. Reference Filepath
Example: **C:\eveFRC_DOC\3\-0006015\-00060153ComCorporation.RTF**
If the Common field is empty, the Last name, First name of the Contact are automatically entered and used.
A Contact record may be used for any subject the user wishes. For instance, 'My Favorites' or 'My Pictures'

- *o'SESAME~FRC* does not restrict a user from manually creating additional Sub-folders.
- Under this sub-folder are stored all electronic file types associated with Contact currently displayed.
- A user may choose to accept, modify or delete the default filename offered by o'SESAME.
  Example: **V:\eve_FRC \E\-0000557\Z10#20090111MO~1640PM;MeetingReq.eml**
  Example: **V:\ eve_FRC \E\-0000557\Z10#20090111MO~1645PM;MedicalRecordReq.doc**
- The use of the underscore _ character is reserved for use by the default o'SESAME 2nd level sub-folder.
  Example: **V:\ eve_FRC \E\-0000557_0000557\PlsCopyMedicalRecordsToDrPhil.pdf**

Functions performed by *command buttons* affiliated with *o'SESAME~FRC*.

- The **[*b*]** button launches a fast, powerful macro that assists the user in finding an application.
  The complete application filepath is automatically stored in the **FRC** Profile field.
  The **[*e*]** button allows the user to manually edit or delete the filepath.
  The **[*Run*]** macro analyzes the filename **prefix** & **suffix** in order to launch the profiled program.
  The **[*Zoom*]** macro does the same for graphic files.

(CPN) Cross-platform Path Name – a Lexicon Recommendation

- **These recommendations are meant for use with standard workstations, laptops and LAN/WAN servers.**
- **Just because a filename you created seems to work, does not mean it can cause problems.**

1. **Avoid using the following characters : / \ . as separators under all Operating Systems.**
2. **Avoid using the following characters + , ; = [] {} * ? " <> | space for a Specific File Name.**
3. The Unix File System is Case Sensitive, but handles pure DOS (8.3) system files with no problems
4. DOS stores all filepaths in UpperCase and the 'space' character is illegal,
   because the 'space' character is used as a separator in Command Line Scripts.
   For that reason the 'sp' character should be considered illegal and avoided in filenames.
   Microsoft ignores these issues with O.S. specific workarounds.
   This and other non-standard techniques are known to cause instability problems with the Windows O.S..
   Windows is Not Case Sensitive and stores all file names in UpperCase.
   Lower-case letters are tracked separately in Windows at the expense of increased storage and CPU overhead.
   Avoid sequentially named files that use identically sequenced filenames. Examples: MyFile01, MyFile0, MyFile03
5. Only a Windows file stored in a root directory can utilize the maximum of 255 characters.
6. Do not use more than one 'dot' in a file name.
7. Do not use *spaces* or *dots* in folder names; their use greatly increases storage and overhead.
8. Do not use more than three (3) characters after the 'dot' in a file name and
   reserve the use of the File Type Extension for its primary function – ***associating*** a file with the application that created it.
9. Large numbers of small files should use all UpperCase letters - again to reduce storage and overhead.
10. Whenever possible, limit individual filenames to a maximum of ***62 characters*** in length;
    since this has been a historical limitation used by many CD read/write programs.

Unix-OS: Unix File Name (UFN) System

- **Avoid using the following characters in a filename**
  | ; , ! @ # $ ( ) < > / \ " ' ` ~ { } [ ] = + & ^ space tab
- **Avoid using these separator characters _ - . :**
- Unix does not care how many parts a filepath or filename has.
- Unix does not recognize or care about file extensions, or what they do.
- Unix filepaths can be up to 256 characters.
- Case Sensitive MYFILE.EXE and myfile.exe are different files under Unix.

Internet Telecommunications

- **Practical size of DNS is typically 255 characters**
- **Practical limit of URL is 2083 characters or less (the limit of Microsoft Explorer)**
- **Internet Servers can deal with much larger values for a 'GET'**
- **Internet names, ip#'s and path conventions are a unique telecomm operating system.**
- **Unix and Linux filenames are typically more Internet compatible than other operating systems.**
- **See Unix-OS above**

IBM-DOS: Dos File Name (DFN) System

- **Avoid using either : or ** because they are used as Directory separators.
- **Avoid using the following characters in a filename / \ : * ? " < > |**
- **Avoid using the 'space' character.** Once it is stored, the 'sp' causes file management problems.

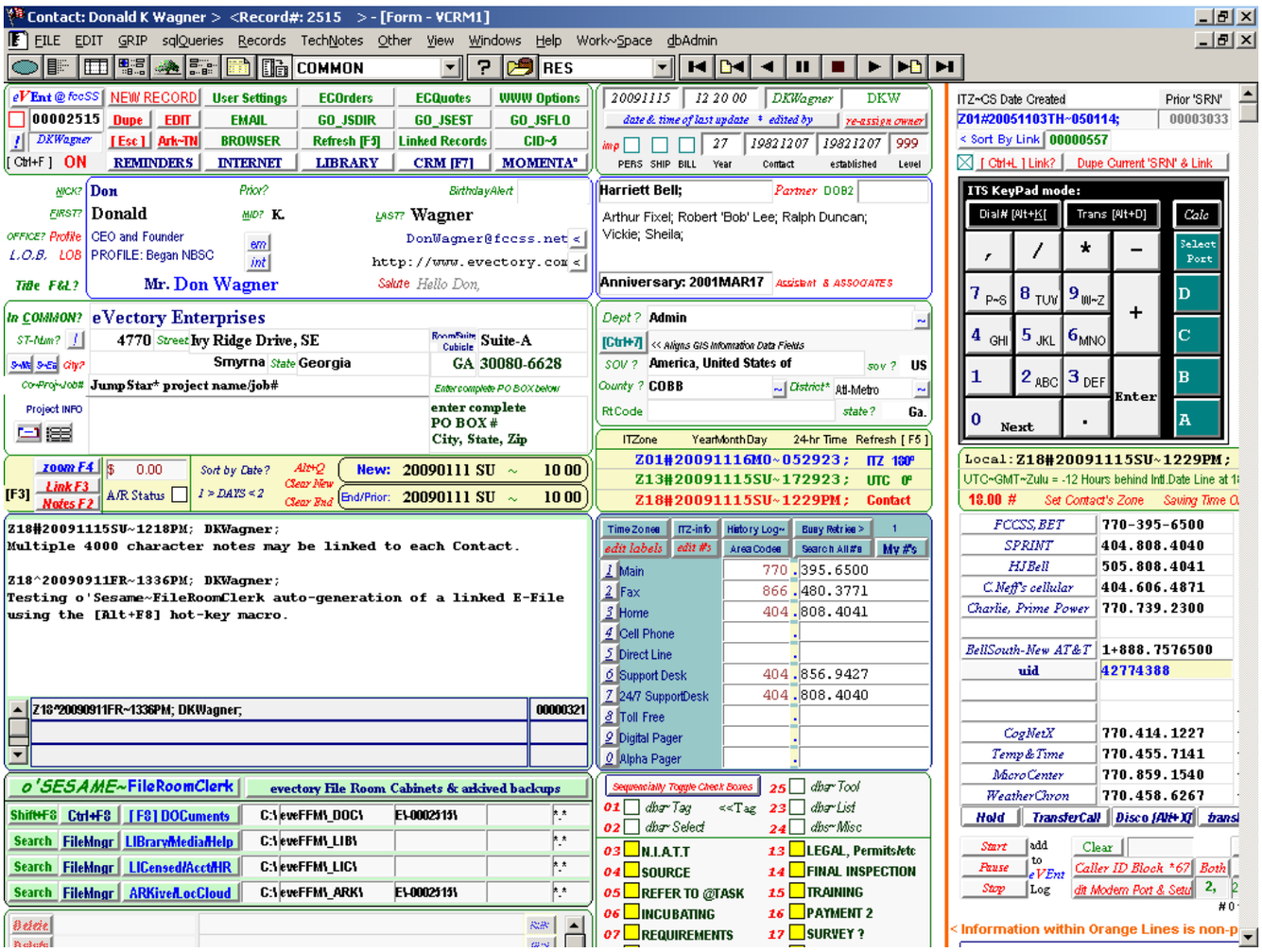
APPLE-OS: Long File Name (LFN) System

- Maximum LFN Path is **260** characters.

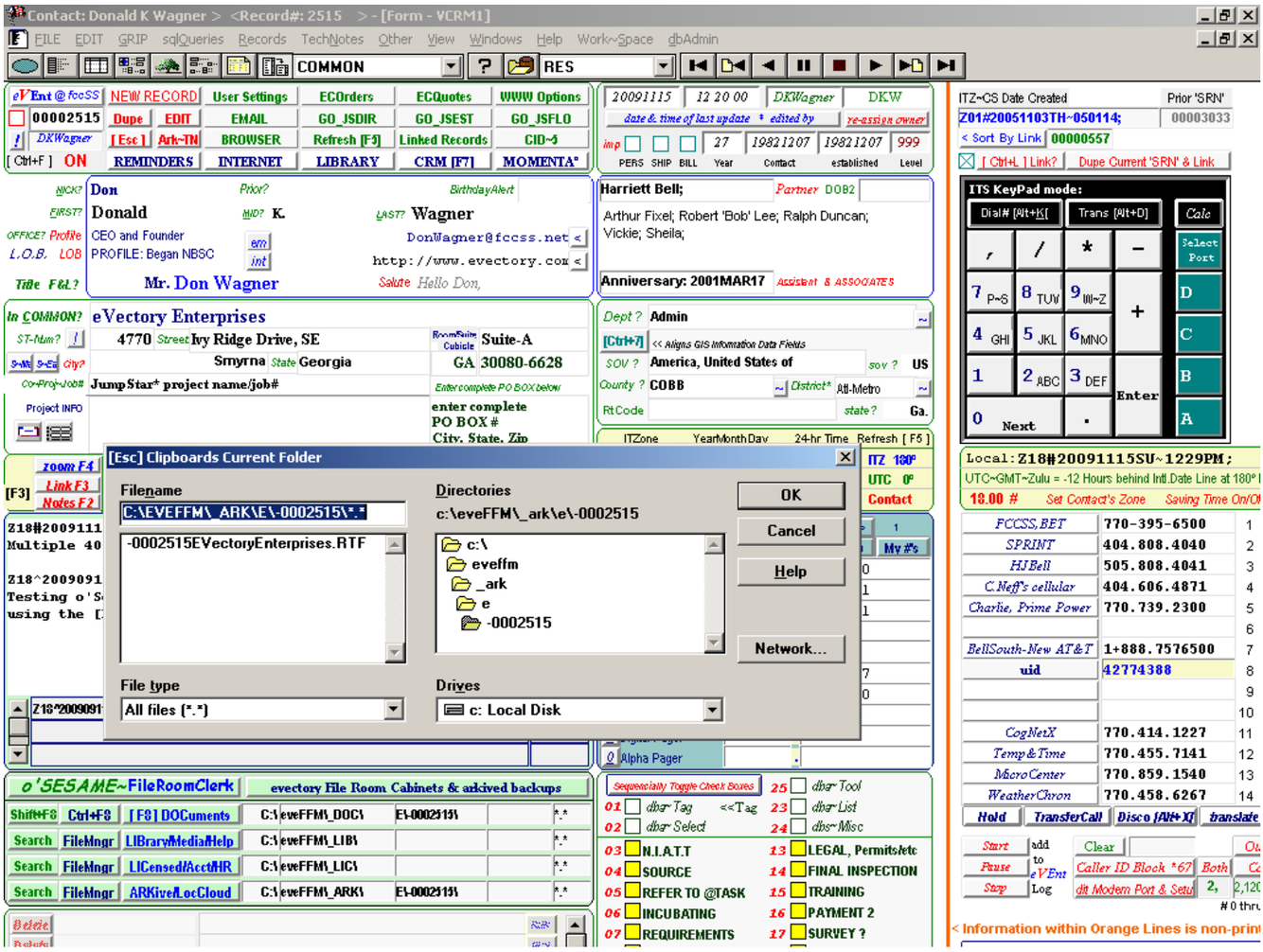
Microsoft's Windows-OS: Long File Name (LFN) System

- **Avoid using either : or ** because they are used as Directory separators.
- Characters Not Allowed Under Windows **/ \ : * ? " < > |**
- Maximum Long Filename Path is 260 characters.
- Maximum length for a single filename is 255 characters. Five characters are reserved for drive path.

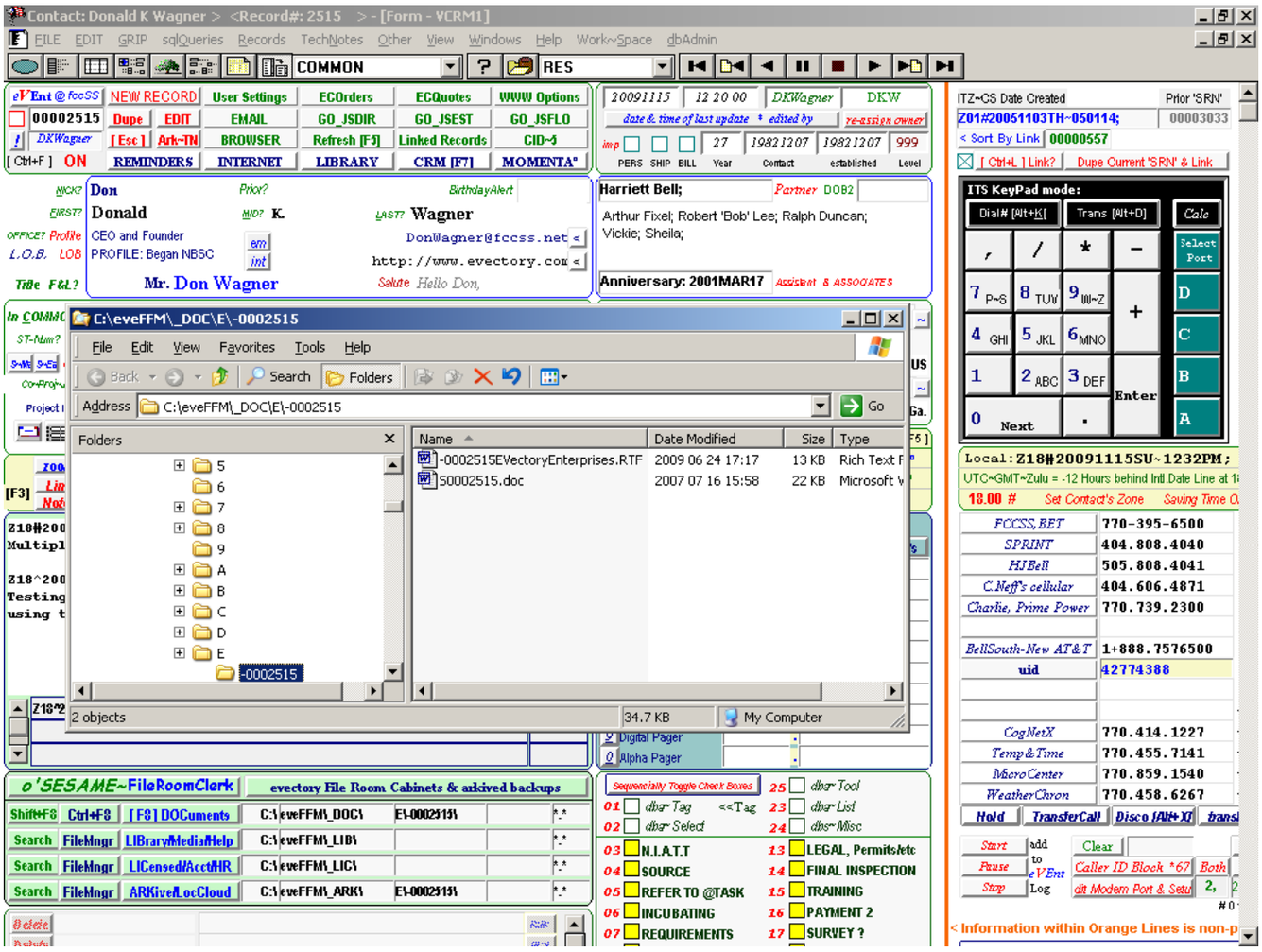
Screen Capture of a noahs*ARK* CRM Contact screen.



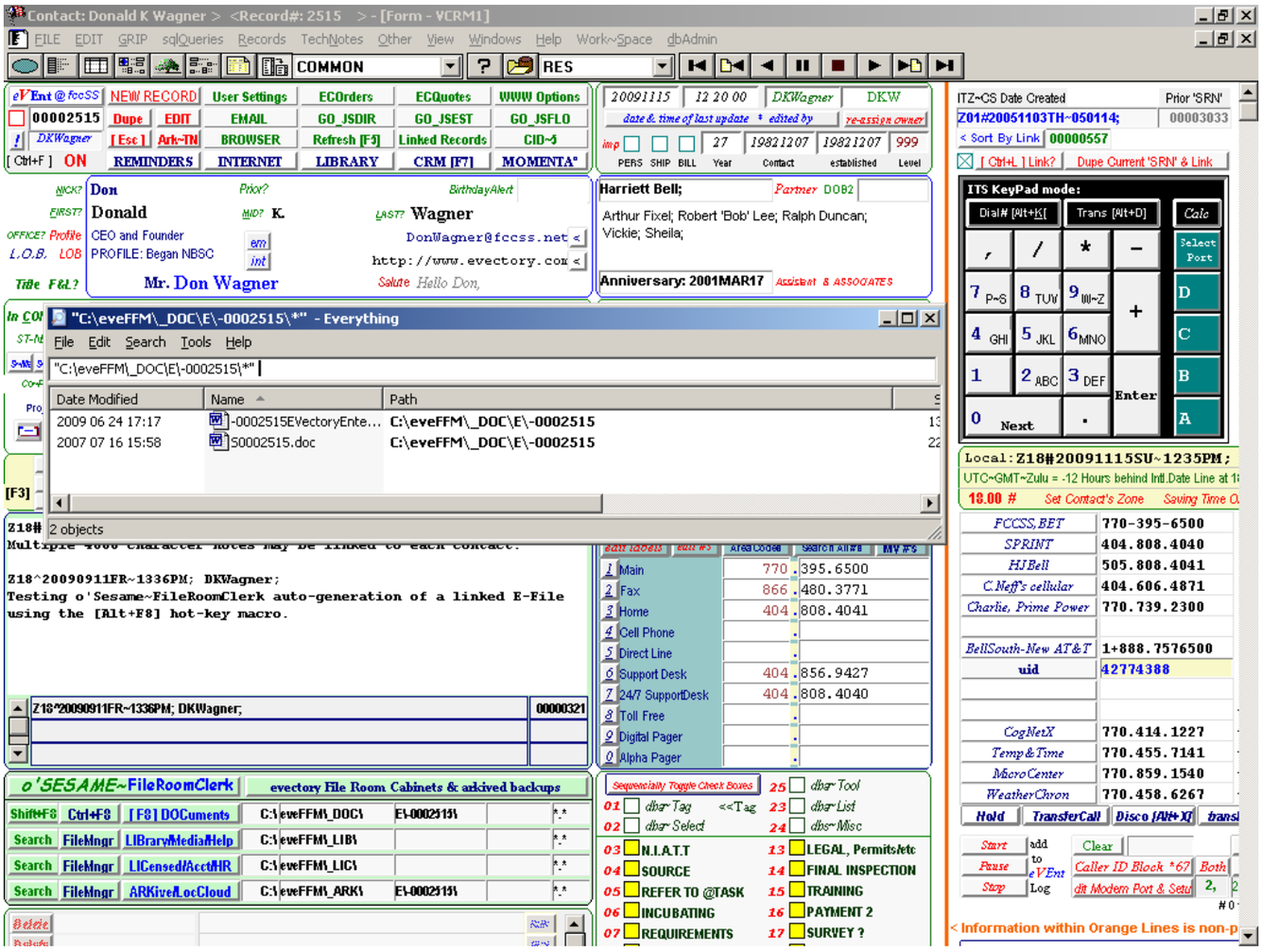
[F8] hot key pops up the FileRoomClerk *FAST* File Manager



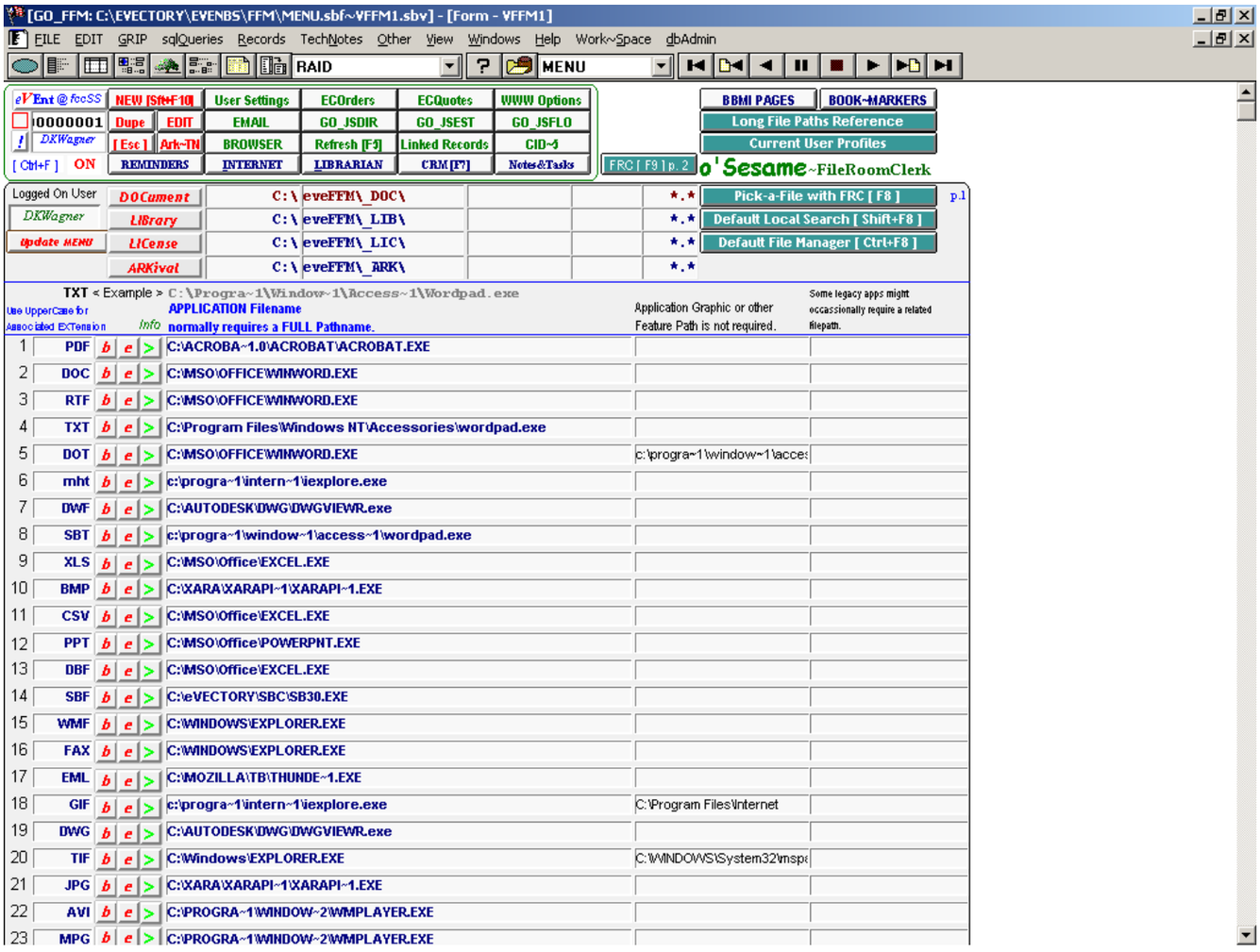
Screen Capture of a Contact screen with MS-Explorer displaying the Contact FileRoomClerk.



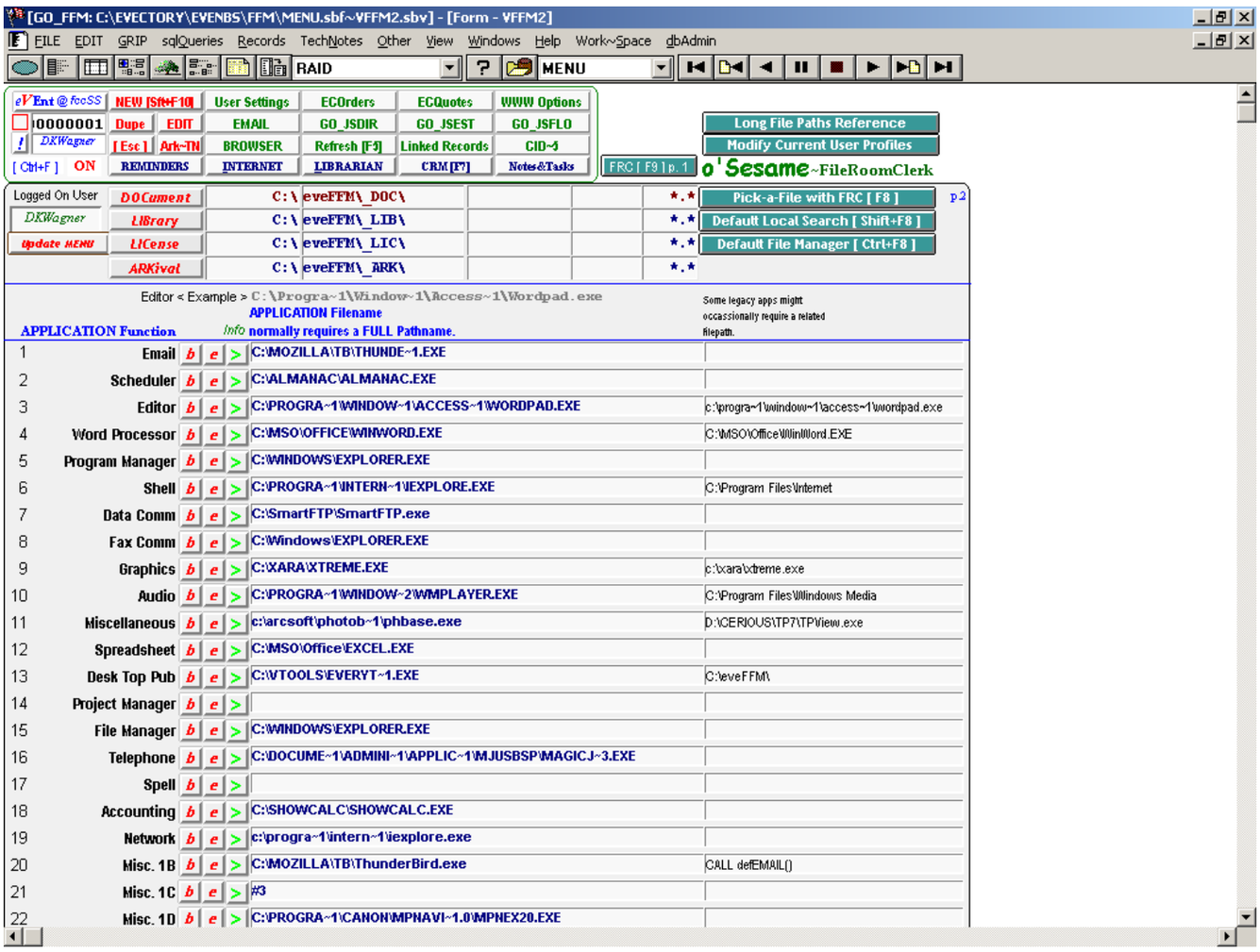
Screen Capture of a Contact screen with VoidTools instantaneous ‘Everything’ NTFS Search Engine displaying FRC Folder



o'SESAME~FileRoomClerk (User Profile setup for application and associated extensions)



o'SESAME~ FileRoomClerk **User Profile** setup for functions and the preferred application relationship.



File System Attributes

Attribute	Bit Code
Read-Only	00000001
Hidden	00000010
System	00000100
Volume Label	00001000
Directory	00010000
Archive	00100000

http://heather.cs.ucdavis.edu/~matloff/UnixAndC/Unix/FileSyst.html

File Systems in Unix

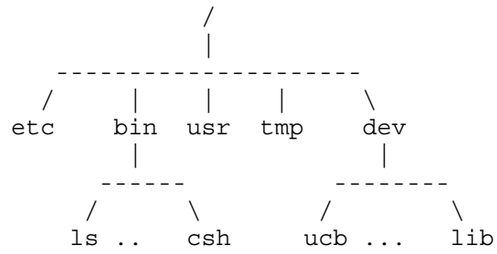
Norman Matloff
Department of Computer Science
University of California at Davis

October 19, 1998

Contents

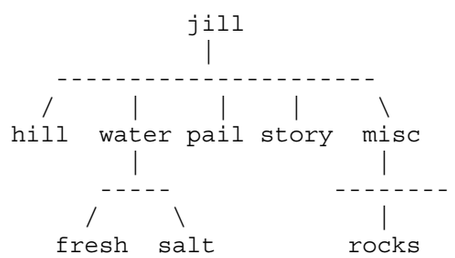
1. Introduction

In Unix, the files are organized into a tree structure with a root named by the character '/'. The first few levels of the tree look like this:



Your own files form a subtree of this tree. For example, in many systems the user files are subdirectories of a directory named `home' within `usr'; if we had users Jack and Jill, for example, Jack's home directory would be /usr/home/jack, and all his files would be within that subtree, and the analogous statement would hold for Jill.

Suppose Jill's directory looks like this:



File names can be given either in relative terms, or with full path names. Look at the file `salt' above. If we are in the directory `water', we can refer to this file as simply

```
salt
```

If we are in the directory above, i.e. the one named `jill', then we must write

```
water/salt
```

If we are in the directory `misc', we can write either

```
../salt
```

or

```
~/water/salt
```

If we are not in any of Jill's directories, we can write

```
~jill/water/salt
```
In any case, the full pathname will work:
```
/usr/home/jill/water/salt
```

2. File Types

There are four types of files in the Unix file system.

2.1 Ordinary Files

An ordinary file may contain text, a program, or other data. It can be either an ASCII file, with each of its bytes being in the numerical range 0 to 127, i.e. in the 7-bit range, or a binary file, whose bytes can be of all possible values 0 to 255, in the 8-bit range.

2.2 Directory Files

Suppose that in the directory x I have a, b and c, and that b is a directory, containing files u and v. Then b can be viewed not only as a directory, containing further files, but also as a file itself. The file b consists of information about the directory b; i.e. the file b has information stating that the directory b has files u and v, how large they are, when they were last modified, etc.[1]

2.3 Device Files

In Unix, physical devices (printers, terminals etc.) are represented as ``files.'' This seems odd at first, but it really makes sense: This way, the same **read()** and **write()** functions used to read and write real files can also be used to read from and write to these devices.

2.4 Link Files

Suppose we have a file X, and type
```
ln X Y
```
If we then run **ls**, it will appear that a new file, Y, has been created, as a copy of X, as if we had typed
```
cp X Y
```
However, the difference is the **cp** does create a new file, while **ln** merely gives an alternate name to an old file. If we make Y using **ln**, then Y is merely a new name for the same physical file X.

3. Obtaining Information About the Files in a Given Directory

The `a' (``all'') and `l' (``long'') options of the **ls** command will give us a lot of information about files in a specified directory (if we don't specify a directory, then the current directory is assumed). Here is a sample output from typing

```
ls -al
drwxr-xr-x  6 ecs4005      1024 Apr 22 13:30 ./
drwxr-xr-x 74 root         1536 Mar 24 12:51 ../
-rw-------  1 ecs4005       188 Apr 13 15:53 .login
-rw-------  1 ecs4005         6 Mar 24 11:29 .logout
-rw-------  1 ecs4005       253 Apr 10 12:50 .xinitrc
-rw-r--r--  1 ecs4005       516 Apr 10 13:00 .twmrc
-rw-r--r--  1 ecs4005      1600 Apr 22 10:59 test2.out
```
The output is separated into six columns:
```
1st column - access permissions (see below)
2nd column - number of file entries (in the case of directory files)
3rd column - owner
4th column - size in bytes
5th column - date and time of last modification
6th column - name
```

4. File Access Control

In Unix, all files are protected under some access control mechanism, so that the owner of a file can deny access of his files to other users. The first column of the long directory list shows the access characteristics of a file, in te form of 10 flags, e.g. drwxr-xr-x.

The meanings of the flags are shown below:

```
Position 1       file type: d (directory)
                            - (ordinary file)
                            l (symbolic link)

Position 2-4     permissions for the owner: r (read)
                                            w (write)
                                            x (execute)

Position 5-7     permissions for other users in the same group

Position 8-10    permissions for all other users
```
Note that a hyphen (`-') denotes lack of the given permission type. For example, r-x would mean that read and execute permission are granted, but not write permission.

In order to remove a file, you must have write permission for it.

In order to view the contents of a directory, i.e. see what files are there, you need read permission for that directory. In order to actually access a file (read from it, write to it, or execute it) in the directory, you need execute permission for the directory.

5. Some File Commands

5.1 chmod

You can use this command to change the access permissions of any file for which you are the owner. The notation used is:

```
u       user (i.e. owner)
g       group
o       others
+       add permission
-       remove permission
r       read
w       write
x       execute
```

As an example, the command

```
chmod ugo+rw .login
```

would add read and write permission for all users to the .login file of the person issuing this command.

In some cases it is useful for a user to deny himself/herself permission to write to a file, e.g. to make sure he/she doesn't remove the file by mistake.

5.2 du and df

The du command displays the sizes in kilobytes of all files in the specified directory, and the total of all those sizes; if no directory is specified, the current directory is assumed.

The df command displays the amount of unused space left in your disk systems.

5.3 diff

This command displays line-by-line differences between two ASCII files. If for example, you have two versions of a C source file but don't remember how the new version differs from the old one, you could type

```
diff oldprog.c newprog.c
```

6. Wild Cards

These will save you a lot of typing!

There are two wild-card characters in Unix, `*' and `?'.

The wildcard `*' will matches with any string of characters. For example,

```
rm *.c
```

would delete all files in the current directory whose names end with `.c'.

The wildcard `?' will match with any single character. For example,

```
rm x?b.c
```

would delete all files whose names consisted of five characters, the first of which was `x' and the last three of which were `b.c'. Example: rm prog?.c will delete all the files (in the current directory) The files x3b.c and xrb.c would be deleted, while the file xuvb.c would not.

In addition,

```
[0-9] matches character from `0' through `9'
[a-z] matches character from `a' through `z'
```

For instance,

```
rm test[1-3].c
```

would remove test1.c, test2.c and test3.c but not test4.c.

Footnotes:

[1] The **ls** command gets its information about the directory b by reading the file b.

Setting Permissions

The chmod command uses as an argument a string which describes the permissions for a file. The permission description can be in the form of a number that is exactly three digits. Each digit of this number is a code for the permissions level of three types of

people that might access this file:

1. Owner (you)
2. Group (a group of other users that you set up)
3. World (anyone else browsing around on the file system)

The value of each digit is set according to what rights each of the types of people listed above have to manipulate that file.

Permissions are set according to numbers. Read is 4. Write is 2. Execute is 1. The sums of these numbers give combinations of these permissions:

- 0 = no permissions whatsoever; this person cannot read, write, or execute the file
- 1 = execute only
- 2 = write only
- 3 = write and execute (1+2)
- 4 = read only
- 5 = read and execute (4+1)
- 6 = read and write (4+2)
- 7 = read and write and execute (4+2+1)

Permissions are given using these digits in a sequence of three: one for owner, one for group, one for world.

Let's look at how I can make it impossible for anyone else to do anything with my apple.txt file but me:

```
$ chmod 700 apple.txt
$
```

If someone else tries to look into apple.txt, they get an error message:

```
$ cat apple.txt
cat: apple.txt: Permission denied
$
```

If I want other people to be able to read apple.txt, I would set the file permissions like this:

```
$ chmod 744 apple.txt
$
```

Detecting File Permissions

You can use the ls command with the -l option to show the file permissions set. For example, for apple.txt, I can do this:

```
$ ls -l apple.txt
-rwxr--r--   1 december december       81 Feb 12 12:45
apple.txt
$
```

The sequence -rwxr--r-- tells the permissions set for the file apple.txt. The first - tells that apple.txt is a file. The next three letters, rwx, show that the owner has read, write, and execute permissions. Then the next three symbols, r--, show that the group permissions are read only. The final three symbols, r--, show that the world permissions are read only.

From Wikipedia

[edit] Comparison of file name limitations

Main article: Comparison of file systems

System	Alphabetic Case Sensitivity	Allowed Character Set	Reserved Characters	Reserved Words	Maximum Length	Comments
MS-DOS FAT	case-insensitive case-destruction	A-Z 0-9 - _	all except allowed		8 + 3	
Commodore 64	case-sensitive case-preservation	any	"		16	Flat filesystem with no subdirs. Space and shift-space are different chars.
Win95 VFAT	case-insensitive	any	|\?*<":>+[]/ control characters		255	
WinXP NTFS	optional	any	|\?*<":>/ control characters	aux, con, prn	255	
OS/2 HPFS	case-insensitive case-preservation	any	|\?*<":>/		254	
Mac OS HFS	case-insensitive case-preservation	any	:		255	Finder is limited to 31 characters
Mac OS HFS+	case-insensitive case-preservation	any	: on disk, in classic Mac OS, and at the Carbon layer in Mac OS X; / at the Unix layer in Mac OS X		255	Mac OS 8.1 - Mac OS X
most UNIX file systems	case-sensitive case-preservation	any except reserved	/ null		255	a leading **.** means ls and file managers will not by default show the file
early UNIX (AT&T)	case-sensitive case-preservation	any	/		14	a leading **.** indicates a "hidden" file
POSIX "Fully portable filenames"[2]	case-sensitive case-preservation	A-Za-z0-9._-	/ null	Filenames to avoid include: a.out, core, .profile, .history, .cshrc	14	hyphen must not be first character
AmigaOS	case-insensitive case-preservation	any	:/"		107	dos.library
Amiga OFS	case-insensitive case-preservation	any	:/"		30	Original File System 1985
Amiga FFS	case-insensitive case-preservation	any	:/"		30	Fast File System 1988
Amiga PFS	case-insensitive case-	any	:/"		255	Professional File System 1993

	preservation				
Amiga SFS	case-insensitive case-preservation	any	:/"	32,000	Smart File System 1998
Amiga FFS2	case-insensitive case-preservation	any	:/"	107	Fast File System 2 2002
BeOS BFS	case-sensitive	UTF-8	/	255	
DEC PDP-11 RT-11	case-insensitive	RADIX-50		6 + 3	Flat filesystem with no subdirs. A full "file specification" includes device, filename and extension (file type) in the format: dev:filnam.ext.
DEC VAX VMS	case-insensitive	A-Z 0-9 _		32 per component; earlier 9 per component; latterly, 255 for a filename and 32 for an extension.	a full "file specification" includes nodename, diskname, directory/ies, filename, extension and version in the format: OURNODE::MYDISK:[THISDIR.THATDIR]FILENAME.EXTENSION;2 Directories can only go 8 levels deep.
ISO 9660	case-insensitive	A-Z 0-9 _ .		255	8 directory levels max (for Level 1 conformance)

[edit] See also