The MiNT distribution kit - version 2.0

1. About:

The MiNT distribution kit is a set of (currently 9) disks that attempt to make it easier to install MiNT onto hard disk. The distribution is maintained by me - Simon Gornall (sjg@unhygienix.ph.kcl.ac.uk) - and can be obtained via anonymous ftp from the one of our DECstations (phlim.ph.kcl.ac.uk). The kit will also be made available to Jeff Weiner at atari.archive.umich.edu

I lay no claim to any of the code contained within this distribution, save the programs on the install disk (setup.prg and install.prg), which are placed into the public domain with the proviso contained within their 'about' dialogue. If anyone is displeased with the manner of this distribution, because I have used their programs without due recognition, or because I have not paid sufficient attention to their distribution conditions, I would ask them to get in touch with me at the above address. If we can't resolve the matter I will remove the program(s) from the distribution kit.

Contained within the distribution kit are:

* MiNT 0.95.
* MinixFS 0.55.
* Gcc 2.2.2 - includes the mbaserel and pcbaserel addressing.
* Bellcore Windows MGR.
* GNU utils including sed, fileutils, etc.
* Init 1.0 - the multiuser extension to MiNT.
* Nroff-compatible manual system similar to un*x
* Many MiNT-aware shells (tcsh, bash, ash, sh, mintshel)

You will require an ST(e) with at least 1Mb ram, a free HD partition of anything up to 14Mb, and ~500k on C:\ You will also require at least a 720k drive. If there is sufficient demand from 360k drive owners then I will try to convert tyhe kit for the smaller disk size.

As and when new software appears, it will be incorporated into the distribution as a patch disk. The interface to both the main installation and the patches will be the same program - install. This is a point-and-click interface for the selection of which programs to install, with the collection of individual related programs / files into subsets. For example, the GCC distribution is in three parts, the GCC header files, the GCC libraries, and the GCC binaries. These may be selected by clicking on the relevant lines in the selection window.

The kit comes on 9 disks currently. There are 8 data disks which contain subsets. The ninth disk is the install disk. It contains 2 programs that aid in the installation of the kit. The first is SETUP, the second INSTALL. The purpose of SETUP is to get the system into a state in which the program INSTALL may be used. Since INSTALL is a general installation program, it knows nothing of what filesystem it is using etc., all this is left to SETUP to get right. There is still the occaisional bug in both setup and install, but they are of the 'annoying' rather than 'critical' variety. For example, setup sometimes minit's a disk twice, but no harm is done. And install sometimes reads garbage for a configuration. A quit and restart solves this.

2. Architecture:

The nine disks are labelled (informatively!) as disk1 ... disk9. Disk 1 is the install disk. It contains the installation programs Setup and Install. The disk also contains necessary programs that get placed in /bin and some for /etc. It additionally installs the TOSwin accessory, and MiNT 0.95 together with the MinixFS filesystem.
The rest of the disks are subset disks. A quick rundown of what is on what disk follows overpage...

Disk 2:	Fileutils:	Eric Smith's port of some of the GNU file utilities. This is almost a mandatory subset, as it contains ls.ttp - how you get a disk directory under a shell.
	Bash:	The GNU Bourne-again shell. (A clone of sh, the bourne shell). Quite a nice version of sh, but quite large as well. It contains extensive online help, which is useful for beginners.
	Tcsh:	The recommended shell for those with lots of memory. This is a clone of csh, the C shell, with lots of additions.Unfortunately it is very big, taking around 350k to run. This will be greatly relieved with the introduction of MiNt 0.96, which allows two identical programs to share the same code segment, so only one copy of the code is kept in memory when two identical programs are being run. A shell is the prime example of a multiply-run program.
Disk 3:	bin1:	This is a collection of binaries I've made over some time. I'm not sure where most of them have come from, so I've lumped them all together. If you intend to use shells, then this is more or less a mandatory install.
Disk 4:	gcc222b:	The GCC 2.2.2 binaries. All the latest stuff that has support for MiNT 0.96's sharable-text stuff as detailed above. Note that no g++ compiler is included.
	gcc222i:	The include files for the GCC compiler. This holds the MiNT includes as well as the standard GCC ones.
Disk 5:	gcc222l:	The GCC 2.2.2 libraries. This includes the 'b' libraries that support the shared-text MiNT 0.96 facility, as well as the normal libraries.
	docs:	Miscellaneous documentation. Lots of it. Includes manual pages, readme files for programs, anything that doesn't fit somewhere else.
Disk6:	futil2:	A more up to date version of the GNU fileutils.
	rg10:	A program + docs that let you run GEM programs from within the tty interface of MiNT. For those who don't use the TOSwin environment.
	tpipe:	A program similar to tee in bin1, that lets you split unix pipes into two.
	sed:	The GNU sed package. It lets you apply editing commands to a file (sed stands for Stream EDitor). A very useful un*x tool.

(More Overleaf) ...

Disk 7:	mnt95utl:	MiNT utilities release 0.95. Provides programs such as 'ps' for reporting processes currently running, 'top' which shows which process is getting what amount of cpu etc.
	akpbin:	Andrew Pratt's utilities. These are similar to the GNU fileutils, but use \ rather than / as a directory separator. I prefer /, so I use GNU utils, and so I don't know whether these work as well as the GNU ones, There are more GNU utils, and they always use /, so I suggest using the GNU utils.
	cpx:	This is a .cpx file which shows what processes are running. You need Atari's Xtensible control panel to run this.
	ksh:	The Korn shell. Sworn by, by many. Alas, similarly sworn at by many. I'm going to sit on the fence and say make up your own mind. I don't think it's as friendly as tcsh, but more friendly than sh. Also, it's not as big as tcsh. A good compromise ?
	man3:	A manual formatting package that uses nroff (a un*x typesetting language) to produce readable manual pages from their standard un*x form. This needs some elements of the bin1 subset to be installed - namely /usr/local/lib/nroff/tmac.an, and /usr/bin/more.ttp
	elvis:	A vi clone. Vi is the original un*x editor (it stands for VIsual), it is actually a line-based editor pretending to be a screen-based editor. For this reason, most people prefer different editors. Still, under proper un*x you sometimes have to use it (for example in the special case vipw, which vi's the passwd file)
	jove:	Jonathon's Own Version of Emacs. A reasonably complete editor, which is my personal choice. Emacs is a screen editor, but some of the keys are a little esoteric. Documentation is included.
Disk 8:	Ash:	Ash is 'A SH clone', which you may or may not wish to install over /bin/sh that is already in /bin. Docs (and a manual page) are included.
	Init:	Init is the multi-user extension to MiNT that makes it behave as though it were a 'proper' un*x box, even down to the boot sequence. The /bin/sh is the one that is installed as standard on the system. If you use multimnt.cnf, then you **must** have installed this subset.
Disk 9:	mgr:	Bellcore windows Manager. This is a windowing system developed for early Sun un*x boxes. It is also available for the Pd un*x clone Linux. Now more or less superseded by TOSwin - which allows GEM programs to be run as well as TOS ones in their own window. Mgr only allows TOS programs to be run, but provides a graphical interface for GEM-like programs. Unfortunately the only program I know of that runs under mgr is mgrterm, which allows the use of a modem, and provides a window in which you can run it.

Well, that's the lot. Any patch disks that come out will start at Patch1 ...

Typically I would expect the user to extract the following subsets...

fileutils, tcsh, bin1, gcc*, sed, futil2, mnt95utl, cpx, man3, jove, and docs. This works well for someone with 4Mb of memory using the TOSwin accessory to do things. With only 2 Mb of memory, I would recommend the above minus tcsh, replacing that with ash, or mintshel. I would still use TOSwin.

For someone who wants to experiment with Multi-user MiNT, you will need to add init to the above. Memory is less tight then, because you'll not be running more than one main shell at a time. Tcsh is a reasonable choice with only 2Mb memory, but don't expect wonders with gcc in that setup!

Other subsets are definately 'optional'. I'm toying with the idea of letting Install remove subsets as well as install them - so you could 'try one out' much easier.

3. FileStructure:

Setup will install a standard un*x file structure on a partition of your choice. This means that in the root directory you will find only the following directories: /tmp, /mgr, /usr, /bin, /etc, and /var. All user-orientated stuff goes into /usr, all system-orientated stuff goes into /etc. /mgr is a special case - it is a windowing system, and deserves a directory in such an important position in the tree. /bin contains basic binaries. By basic I mean the binaries that the system cannot afford to be without. /var normally contains things like the admin stuff, the spool directories, etc.
/tmp is a temporary directory that exists on ALL un*x systems. Under MinixFS you will find that it is set to be 'sticky'. This means that anyone can write into it, but you can only delete your own files, not anyone elses. Doing an 'ls -l /tmp' will reveal...

rwxrwxrwt 2 root wheel /tmp

... which means that everyone has Read, Write, and eXecute access, the sticky bit is set (the 't'). The owner is root, the group ownership is 'wheel' (standard root group), and the directory is called /tmp.

The next most important directory is /usr. This is the largest directory on most un*x clones. It contains /usr/bin - where the vast majority of binaries will go, /usr/lib - where all the gcc libraries are kept, /usr/include - where all the include files for gcc are kept, /usr/ucb - where all un*x 'contributed' software that has made it into the standard distribution goes, /usr/local - where the stuff local to this machine is kept; normally there are subdirectories of bin,lib, include, man in /usr/local, /usr also contains /usr/man - where all the manual pages are kept in their own subdirectories of man1, man2...man8. That also has to be the longest sentence in this document :-).

Your own programs/data ought to go in your 'home' directory of /usr/users/<username>. In here you should have a bin directory, and whatever other directory structure you like. Your home directory is the only place I guarantee that future patches will not touch. All other directories on the system I reserve the right to modify. There shouldn't be a major problem here, because patches will always tell you what they will do in advance (in the information window) but be warned anyway that outside /usr/users/<username> anything is fair game!

The un*x filesystem is one which C compilers tend to understand automagically. I don't know if this is the case with the GCC compiler available (Howard ?), but it would be nice if it was. Setup will produce environment variables of GNULIB and GNUINC etc. anyway. You will, however, have to be on drive U: (the unified filesystem) for this to work. You will find that drive F: (for example) is accessible as U:/f/.

4. Unpacking the disks:

The disks all come as .zoo files. You will find that there are 9 .zoo files named disk1.zoo, disk2.zoo, ..., disk9.zoo. They are compressed images of 720k floppy disks. To install using them, you must first 'un-zoo' them. The method I suggest is to un-zoo them into an empty directory on your hard disk (call it 'floppy' or something), then copy all the files from 'floppy' to your floppy disk. You will find binaries of zoo for DECstations and SPARCstations in with the distribution kit so you can do it on the workstation if you have access to a floppy drive on your network - I believe this is fairly common. Note that the floppy in DECstation 5000's is a 2.88Mb floppy and cannot write 720k disks!

The command for zoo that will un-zoo them properly is to type ...

zoo x// <filename>[.zoo]

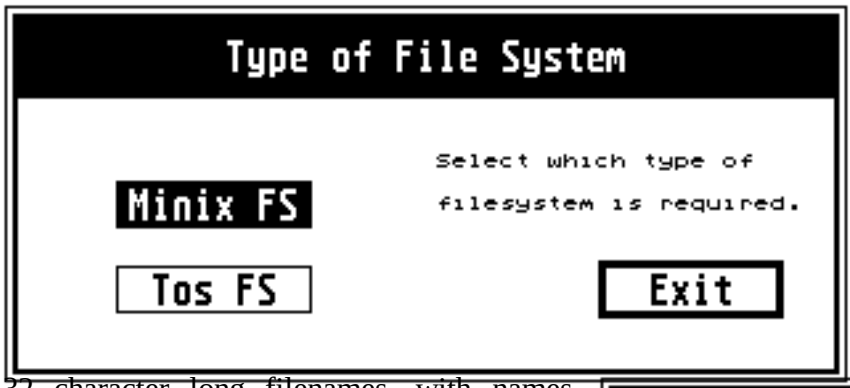
... which will create subdirectories as needed. The compaction used on all of these subsets is the high-performance compression technique, which is actually slightly **faster** to decompact than the normal technique. It is comparable to the -lh5 option on the latest lharc's available.

Typically (excluding disk1.zoo), you will find that the majority of space within the subsets is taken by more .zoo files, so there will be little compression. Still, it is easier to move 1 file around than the 30 or so that are on disk1 for example.

5. setup:

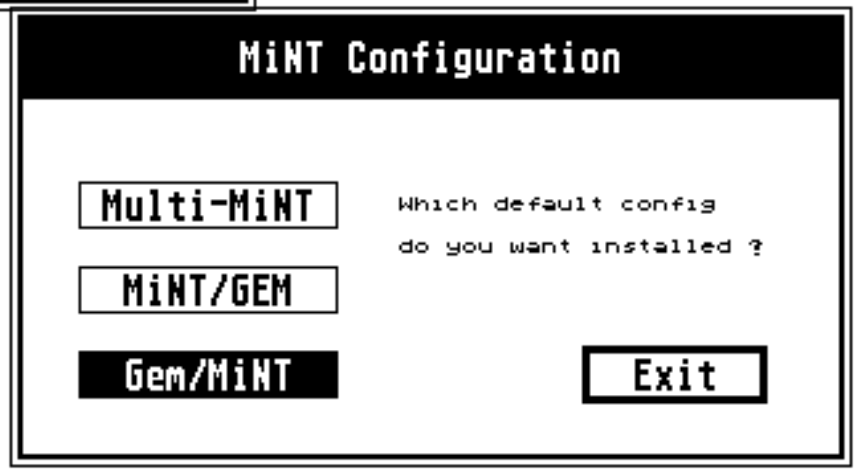
The first program to be run is Setup.prg on the installation disk. This is what will initialise a MinixFS partition, and create a skeleton directory structure into which Install may put programs. Setup has a few options which will be explained below.

There are 4 menu bar entries. The first is the Desk entry common to all Gem programs. The second is the File menu where there are 4 options, the first is "Set Root", which lets you set which partition to install MiNT onto. The distribution kit will create a un*x-like file structure starting from the partition you specify. Make sure you select the root directory of a partition.
Secondly, there is "Save Info". This saves a small configuration file to the floppy disk which will be read on startup so that you don't have to keep selecting which options. Currently it stores the location of zoo.ttp and the root of the filesystem to install onto.
The Third entry is "Install Configuration", which will fire up the installation procedure. There will be a warning given, because this is such a dangerous procedure.
The fourth and last entry under "File" is to quit the program.



the third menu is "Options", which contains entries allowing you to select which type of MiNT configuration you prefer, and which sort of filesystem you want to run on.

The dialogue opposite appears when you click on "Select Filesystem". The TOS filesystem is the normal Atari filesystem, with 8+3 filenames etc. If, however, you choose the (default) minix filesystem, you can have up to 32 character long filenames, with names such as 'filename.tar.Z' being valid. UPPER and lower case filenames are distinct, and filenames beginning with a '.' (full stop) are considered by ls to be hidden - they can be seen using ls -a, but for most of the time they are not reported. This tends to be used for resource files under un*x.
The disadvantages of minixFS are that MiNT will need to be running before you can access any data on that partition. The system makes sure that TOS will not disturb the minixFS when MiNT is not running, so no data will be lost.



The other choice is which default configuration you want installed. Actually the installation will build the three configurations described, so this option is no longer as important. As a default it will install the option that lets GEM and TOSwin run, so that you have full access to GEM, and can run TOS programs in a window on the desktop, with cut and paste between a global clipboard. Anyone who has used xterm will feel at home with TOSwin running a shell. The default is called gemmint.cnf (runs GEM, but lets you access MiNT), the others are mintgem.cnf (which runs MiNT and starts you up in a shell as user root. to run gem from this stage you simply run the program gem.ttp which will start gem and keep MiNT running. Alternatively you can press CTRL-D in a shell (or type 'exit') and the shell will terminate, so will MiNT, and GEM will start as if MiNT had never happened. The third choice is multimnt.cnf which starts MiNT up into a multiuser version, for this you must install the 'init' subset.See the later chapter entitled Multi-User MiNT.

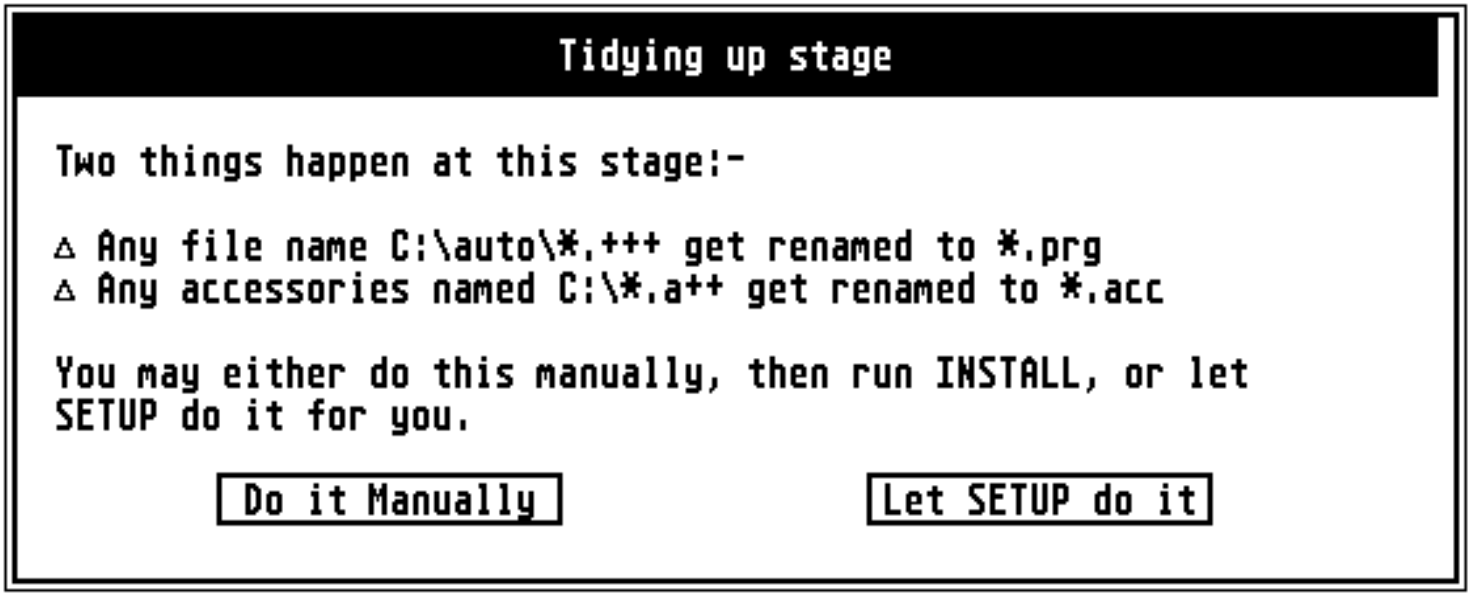
Clicking on "Install config" will start the thing going. The program tells you what it is going to do, with a dialogue box for all the important parts. The first major dialogue is when Setup reports that it is about to mess with the filenames on your C:\ partition. The reason for doing this is that there are some accessories and auto-programs which interfere with MiNT. The program will allow you to quit if you wish to do some renaming yourself (eg: if you already have a C:\mint or C:\bin.sjg directory, you will want to change them, because they will otherwise be lost). See the dialogue below.

Once you have let Setup do its stuff it will start to copy files from A:\ and will create the three configuration files outlined above (multimnt.cnf, gemmint.cnf & mintgem.cnf). It will also ask you for a username and real name to go into the password file. This is done so that on a minixFS system, the command "ls -l" will return the correct ownerships of files. It is really only of use in the multimint environment, since then you create files as a particular user, whereas in both other environments, you will have effective id of root - the system manager. The dialogue will look like that below...

Please replace my name and username with ones of your own. I use sjg because that is my login at work, and I've got used to it. Many people prefer their christian name as their username - all down to personal taste. The kit will create a 'home' directory of /usr/users/<username> for you.

Once this has been done, you will have to reset the ST(e), so that MiNT can be booted, and the next stage of Setup started by again clicking on "Install config". This stage (optionally, but recommended) applies a minixFS to the selected partition you have chosen. You will then have to reset the ST(e) a second time so that the hard drive software can recognise a minixFS partition. Some HD software has no trouble, other (notably ICD) software tends to corrupt the disk. It probably doesn't expect a hard disk to undergo a media change! Once reset, run Setup once more, and "Install config" for the last time. A skeleton disk directory will be created, and some files will be copied into /bin, /etc, & /usr/bin. On a minixFS partition they will be protected as well. On a TOS partition, Setup will report that it is "Protecting the filesystem", but it is designed to fail quietly, and no harm will be done.

After this has been done, you will have to reset the ST for (hopefully) the last time, so that the symbolic links that have been installed will be linked to the directories just created. A dialogue will appear like that below...

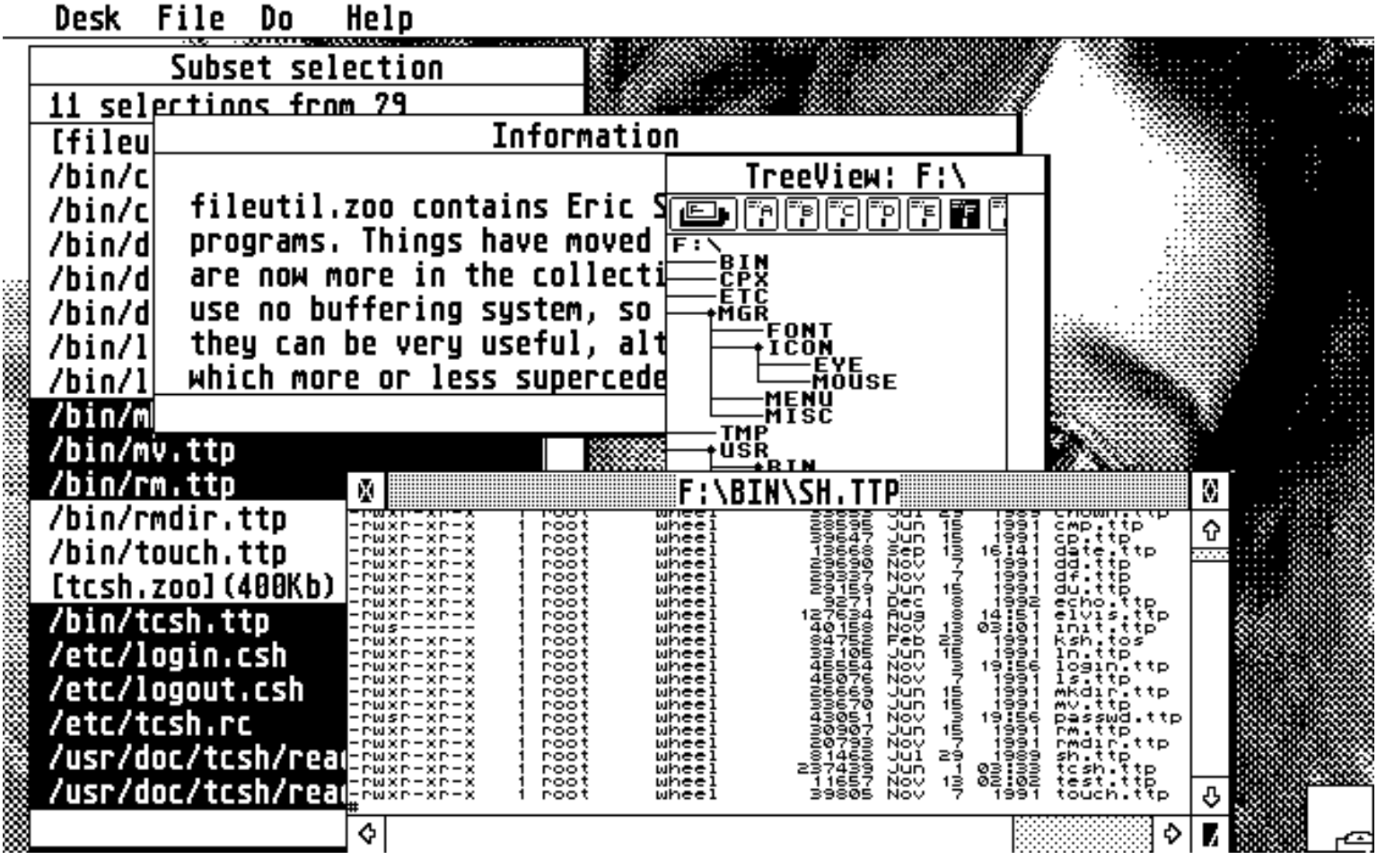


So, you now ought to be ready to run Install, and copy some or all of the installation files to your disk.

6. Install:

Install.prg is a general purpose installation program that unpacks from .zoo archives onto a given partition using a configuration file that is named <anything>.sys, and which will recognise bourne shell scripts with the extensions of <anything>.pre and <anything>.pst. It will pipe these scripts through to a bourne shell that is assumed to be located in /bin on the partition on which installation is taking place. See appendix A for the details of the configuration file.

In general, the desktop for Install won't look something like that below...



... but it could do. The parts of the desktop generated by Install are the Information window, and the Subset window. The File drop-down menu has several options similar to Setup, but it has one entitled "Load Config", which will load in the <anything>.sys file from the selected drive (A: or B:). This initially appears in the Subset window as a set of filenames, one per line, none of them selected. You will notice that some of these lines are of the form /xxx/yyy/zzz/filename and some are of the form [subset name.zoo](size when installed). Clicking on a filename line will select that filename to be extracted. Clicking on a subset name will select ALL the files in that subset to be extracted.

To actually do the extraction, you must select "Install configuration" from the 'Do' menu. If you see that the "Run Script" menu entry is selectable (not greyed out), then there is a script on disk that must be run. Clicking on this menu entry produces a fileselect box which will list the scripts on the disk. You can see when to run it (before or after installing a subset) by the filename. If the filename is <anything>.pre then it MUST be run BEFORE installing the subset that has the same name as the script file. If the filename is <anything>.pst, then it MUST be run AFTER installing the subset with the same name. For example, if the subset is called init.zoo, and there is a file called init.pst, then init.pst must be run after installing init.zoo.

The last menu is entitled 'Help', and will give you brief help on all the functions in the program. The first entry in the Help menu will open a window called "Information", which will give brief details on the subsets that are on the floppy disk.

Warning: Install.prg knows nothing about what you are installing, and some of the files can be **very** dangerous. The installation kit relies heavily on programs such as /bin/sh, /bin/cp, /bin/rm, /bin/rmdir, /bin/mkdir, /bin/chmod and /bin/chown. There are several varieties of these programs knocking around. The ones that are installed onto the system by Setup will work fine. If you overwrite these with others of the same name but without all the functionality, then Install may not work as required. Be **very** careful with programs that go into /bin! Having said that, if you keep the install disk handy, you can always copy stuff back onto your un*x partition. The files in question are in /bin on the floppy disk.

7. In Use:

Once you have installed all you want to install onto your hard disk, you can reboot, and MiNT ought to start up with
TOSwin starting as an accessory, and the GEM desktop starting up. At this stage you will probably want to edit some of the configuration files for the shell that you have chosen to use. The main thing is the PATH environment variable. Under sh (or any of its' clones - bash, ash etc), you will want something like this in a file called /etc/profile ...

```
PATH=/bin:/usr/bin:/usr/users/<username>/bin:. ; export PATH
HOME = U:/usr/users/<username>
cd U:/
cd $HOME
```

whereas if you are running tcsh, you will want something like this in a file called /etc/login.csh

```
setenv PATH /bin:/usr/bin:/usr/users/<username>/bin:.
setenv HOME /usr/users/<username>
cd U:/
cd $HOME
```

I would have put this into the mint.cnf file, but I can't put ':' in a setenv statement there, it expands into /dev/, so the PATH is completely wrong.

I've now modified Setup to create the files above automatically, but I've left the reference so you know what config files to alter. Tcsh also reads a file called tcsh.rc in your home directory, and sh reads a file called .profile in your home directory.

I've noticed that sh doesn't seem to recognise that it is a login shell, so you may need to type ". /etc/profile" for the shell to pick up the file.

At this stage, you ought to be able to select 'Open Standard Window' and run a shell - try typing 'ls /bin', and a directory of /bin ought to appear. Or try 'gcc file.c', and Gcc ought to fire up.

For those who are not using TOSwin, it is slightly more arduos than selecting a different shell from the fileselector to change your shell. Under MultiMint, you'll need to change your password entry from /bin/sh.ttp to /bin/tcsh.ttp or whatever you are using. Under the mintgem.cnf state, you need to change the INIT= line at the end of the file. Again, change it to INIT=/bin/tcsh.ttp (for example) to change your shell.

That's about it. Good luck, and mail me if I've made any major mistakes (or even minor ones.)

Simon.

Appendix A:

The configuration file for install.prg looks something like the following...

```
[zoo file on disk] (any comment inside brackets)                uid gid mode
#
# Any number of
# lines that begin with a '#' character which form the comments that will be put into the information
# window.
#
#
/xxx/yyy/zzz                                                    uid gid mode
/qqq/www
/eee/rrr
/ttt/yyy/yyy/yyy/uuu/iii
[next zoo file ad nauseum]()                                    uid gid mode
```

The first line must be a subset line. That is, it must contain all of the following elements

[name of an archive on the floppy in square brackets] followed by optional spaces
(a comment. This is usually used for the unarchived size of the subset) followed by at least one space
uid is a number that determines the default uid (user- id)for all the files in this subset.
gid - same for the group-id
mode - the default file protections as settable by chmod

Following lines are preceded by a hash sign (#), and make up the description that will be placed in the information window.

Immediately after the lines beginning with '#' comes the list of files that may be extracted from the archive. Each line consists of a file to be extracted relative to / - the root of the partition in which to extract -, and optional uid, gid, and mode fields. If any of the uid/gid/mode fields are present, all must be. If a file does not have a uid, gid, & mode fields, it will be given the default values for the subset.