A guide to the MiNT Distribution Kit

Contents

The MiNT Distribution kit

This document was prepared using an Atari STe and Caligrapher Professional

1. About...

The MiNT Distribution Kit (hereafter the MDK) is an attempt to make the installation of MiNT itself, combined with a fair number of utilities far easier. A secondary aim is to provide an ordered filesystem, with everything in its *proper* place. Since the vast majority of the utilities and programs have been lifted from the Un*x world, this implies a Un*x-like file organisation. Indeed, the un*x system is a particularly ordered affair, with all executables going in /bin or /usr/bin (for example), depending upon importance. The use of a Un*x file organisation is also beneficial in that a lot of people on the net will be familiar with that layout.

The MDK provides for the installation of all the Un*x'y programs that make Un*x an attractive operating system - programs such as mgr, Gcc 2.3.3, bison, sed, grep, fileutils, txtutils, shellutils; added to that is the ability to use a Un*x-like filesystem, with long filenames and case distinction eg: *thisfile.tar.Z* which is highly advised. On top of this is the ability to install the multi-user extension to MiNT that turns your ST(e)/TT into a multi-user multi-tasking machine.

Anyway, what the MDK boils down to is a pair of programs called SETUP and INSTALL which do all the hard work of getting a partition of your hard drive prepared for the MDK, and of installing the system itself onto the partition, and into C:/auto, and C:/mint. The MDK is organised into a set of *subsets* that contain related programs / documentation / etc. INSTALL will then let you pick and choose from the subsets on your disks. The updates to the system also come in the form of subsets, so you can use INSTALL to place them onto your partition as well.

What I ought to get out into the open straight away is that I lay claim to very little of this kit. The installation and setting up programs are mine (Oh don't I know it!) and a few of the subsets have been ported by me to MiNT, but the vast majority of the programs etc. that you will install have been coded / ported by others. Neither, actually, am I necessarily an expert on these programs. I am sufficiently familiar with most of them to be able to give advice, but bug-reports should go to the authors as well as me. Them so they can fix it, me so I can try to work around it.

Another point that ought to be made is that the kit will not work on some TOS's - notably TOS 1.0, and a few of the more exotic TOS's have problems too. The problem with TOS 1.0 is due to the Pexec() call being very buggy, which is used extensively in the kit. There are no guarantees whatsoever if you use this kit, you do so at your own risk, I will not be held liable for any damage / loss / nuclear strikes / etc. (I think this is silly, you're all perfectly nice and reasonable people, aren't you ?)

2. Requirements.

The MDK has some (particularly ?) extreme requirements... The following are necessary:

- A free partition on your hard drive of at least 9Mb which has been formatted to a sector size of 512 bytes. 1024-byte sectors *might* work, but no guarantees. For most of us, any partition less than 16Mb will be fine, but >16Mb, you'll need to have HD software that handles this, or alter the drive with a disk editor. I am told that the ICD software definitely does let you have large 512-byte partitions.

- At least 1Mb free on C:/ and an auto folder on that disk. I assume that C: is your boot partition, if it isn't, then you'll have to compensate by hand-copying all the stuff the MDK puts into C:/auto onto your boot partition. Note that only about 200k of space is eventually used on C:. but some temporary workspace is needed.

- At least 2Mb of memory. At least, you might be able to get away with 1Mb, but I've not tried it, and the kit itself is fairly memory-intensive. You really ought to have 2Mb to get anything out of MiNT anyway, and 4Mb is much nicer. Gcc needs 4Mb to do anything useful

3. What is where.

There are 9 disks in the base installation, labelled (informatively!) as disk1, disk2, ... disk9.zoo. Disk1 is the install disk, with the other 8 being data disks from which you load subsets. A quick rundown of what is on the 8 disks follows...

Disk2:	Fileutils:	Eric Smiths port of some of the GNU file utilities. At least 1 version of the GNU file utils is mandatory.
	Bash:	The GNU Bourne-Again SHell. (A clone of sh, the bourne shell). A nice version of sh, with extensive online help. I don't recommend sh for those without some experience of Un*x though.
	Tcsh:	A clone of csh with lots of additions. The recommended shell if you've enough memory.
Disk3:	bin1:	A collection of binaries I've made over some time. Mandatory, IMHO.
Disk4:	gcc222b:	The Gcc 2.2.2 binaries. This subset contains part of the GCC C compiler.
	gcc222i:	The MiNT-aware include files for gcc 2.2.2
Disk5:	gcc222l:	The libraries for gcc 2.2.2
	docs:	Miscellaneous documentation. Includes manual pages, readme files for programs etc.
Disk6:	futil2:	A more up-to-date version of the GNU file utils. Recommended over the subset on disk2.
	rg10:	Allows the running of GEM programs from a command-line screen. For those who don't wish to use TOSWIN.(More about TOSWIN later).
	tpipe:	A program similar to tee in bin1 that lets you split Un*x pipes into 2.
	sed:	The GNU sed package. Allows the application of editing commands to a file. (sed stands for Stream EDitor).
Disk7:	mnt95utl:	MiNT utilities v0.95. Provides programs such as 'ps' to get a list of running processes etc. Again, almost mandatory IMHO.
	akpbin:	Andrew Pratts utilities. Similar to the GNU ones, but use \ rather than / as a dir separator. I prefer /, but its your choice...
	cpx:	This is a .cpx file for showing what processes are running. You nee xcontrol.acc to make any use of this.
	ksh:	Another shell, the Korn shell. More friendly than sh, less friendly than tcsh. Takes less memory than tcsh, more than sh.
	man3:	A manual formatting package that uses nroff to format Un*x-style manual pages. You need tmac.an and more.ttp from the bin1 subset to use this.
	elvis:	A vi-clone. You'll either love it or hate it. Vi isn't the friendliest editor around, but it is very powerful. Almost as powerful as mainframe emacs.
	jove:	Jonathon's Own Version of Emacs. The nicest editor for Un*x IMHO.
Disk8:	ash:	A SH clone. Another MiNT-aware shell you may prefer over the default sh.
	init:	The multi-user extension to MiNT. You must install this if you want multi-user MiNT.
Disk9:	mgr:	The Bellcore Windows Manager. A windowing system developed for early Sun workstations. Also available for the pd Un*x clone Linux. Now more or less superseded by TOSWIN which is installed as standard. Doesn't allow GEM programs to be run, unlike TOSWIN.

4. Additions to the system.

The upgrade subsets are detailed below. Those marked as ‘shareable’ share the memory they use for program space when you invoke more than one copy of them running at the same time. This can be highly useful for shells, for example.

All of these should be considered to be upgrades, and not as subsets in their own right. In general they will depend on older stuff being already in place. You will find a file called 0readme in the ‘Updates’ and ‘Shareable’ directories which details the chronological order in which the subsets were released.

The following are upgrades:

gcc231	Gcc version 2.3.1 binaries.
incObj25	The MiNT includes and libraries patchlevel 25.
compress	The ‘compress’ Lempel-Ziv-Welch compression utility.
man4	The upgrade to the manual paging system to avoid the MiNT library bug.
sed113	Version 1.1.3 of GNU sed.
init1	The patch to a couple of programs in the init1 subset
shlutil1	The GNU shell utilities compiled with the latest Gcc
mfsutl	A patch for the minixfs system for certain circumstances.

The following subsets are shareable subsets. They may be used with any version of MiNT, but it takes MiNT 0.96 or greater for the shareable effect to manifest. I cannot distribute in binary form any version of MiNT > 0.95 because of licensing restrictions. Nevertheless, eventually a patch to take MiNT to version >0.96 will appear, so these are useful subsets.

bgcc233	Gcc version 2.3.3
bgrep	GNU grep and egrep.
bpatch	Larry Wall’s patch program
btxtutil	The GNU text utilities
bbison	The GNU version of Yacc.

This is a complete list to date. However, the kit will continue to evolve, as will the programs available for it. As time goes on it is my intention to incorporate the newer versions of the same files into the older ones, so there will be less duplication.

5. File Structure.

Setup will install a standard Un*x-style file structure onto the partition you have told it. This means that in the root directory you will find only the following directories: /etc, /bin, /usr, /tmp, /var, and /mgr. All system-specific stuff (boot files, host tables, etc.) goes into /etc. All the *critical* binaries go into /bin - by critical, I mean those without which the system cannot survive./usr holds the user-orientated stuff (including the actual user files, and the programs / data that the user will use from day to day). /tmp is a directory that always exists on Un*x machines, where anyone may write, but may only delete what (s)he has written. Mainly used by compilers that need temporary work space etc. /var contains admin. stuff and spool directories. /mgr is unique to the MDK, and has such an important position in the filesystem because it is an entire windowing system.

Assuming that you are going to use the Un*x-like filing system Minixfs, you will find that files have different levels of privilege. For example, doing an 'ls -l' operation on /tmp will reveal:

rwxrwxrwt	2	root	wheel	/tmp

... the rwxrwxrwt refers to the privilege levels set on the file. the first 3 refer to the owner of the file (in this case 'root'), the second 3 refer to the user group that the file is a member of (here 'wheel'), and the third 3 refer to the rest of the world. So in the above example, the user, the group, and the world have Read, Write, and eXecute access to the directory. (Execute access on a directory means you can go into it). The 't' at the end rather than an 'x' means the sTicky bit is set on the directory. For a directory this means that the files inside are sticky, ie: only their owner may remove them. For a normal file (very rare) it means that the file will not be swapped out on a virtual memory machine - irrelevant to the ST. You may occasionally see something like:

	rwsr-xr-x	1	root	wheel	filename
or	rwxr-sr-x	1	root	wheel	filename

... here a '-' means that the relevant privilege is unavailable (so in the above, only root could write to the file), but the 's' is somewhat more special. It is the 'set-uid' bit (in the first case) or the 'set-gid' bit. It means that on execution of the file, the user-id (group-id) will be assigned to the running process. To explain that:

In case 1 above, if user 'sjg' of group 'users' ran the program 'filename', the program would run as user-id 'root' and group-id 'users'.

In case 2 above, if user 'sjg' of group 'users' ran the program 'filename', the program would run as user-id 'sjg' and group-id 'wheel'.

6. Placement

The positioning of a given sort of file in a given position is the hallmark of the Un*x filesystem. The following is a list of which sort of things go in what directories:

/bin	system-critical binaries. Things like 'ls', 'more', shells etc.
/usr/bin	the vast majority of binaries (programs)
/usr/lib	system libraries (eg: C compiler libraries)
/usr/include	include files for the C compiler
/usr/ucb	'contributed' Un*x software. Only a printer daemon at present.
/usr/man	where all the manual pages are kept. Divided into sections.
/usr/doc	where the documentation that isn't a manual page is.
/usr/local	contains local-to-this-machine dirs of bin, man, include, lib, etc.
/usr/users/x	the 'home' directory for user 'x'. I guarantee to leave this alone.

Your home directory is safe, but anywhere else in the filestructure is fair game to any patches or updates that I make available.

7. Making a start- Unpacking the disks.

The disks all come in .zoo format. You will find that there are 9 .zoo files named disk1.zoo, disk2.zoo, ..., disk9.zoo. These files are compressed forms of 9 floppy disks that are needed for installation. They are compressed for ease of transportation onto your (Mega) ST(e)/TT at home. When you are ready to begin installation, you must first create the 9 disks from the 9 .zoo files by unzoo-ing them. I suggest copying the .zoo file to an empty directory on your HD, (call it 'mdk' or something), unzooing them there, and then copying the files back to a floppy disk. SETUP and INSTALL currently work from floppy, not HD, so you'll have to have them on a floppy I'm afraid.

You will find binaries for zoo on the ftp machine (in the Zoo directory!) for SPARCstations, DECstations, and for the ST. It is of course possible to do this initial unzoo-ing on the Un*x box if you have access to a floppy drive on your network - I believe this is quite common. Note that the floppy on a DS5000/25 is a 2.88 Mb floppy and cannot handle writing to 720k disks.

The command for Zoo that will unzoo them properly is:

```
zoo x// filename[.zoo]
```

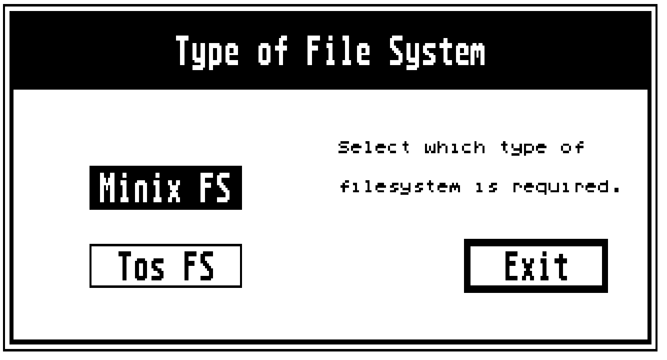
... which will create subdirectories as necessary. The compaction used on all the subsets is the high-compression switch, which is actually slightly *faster* to unpack. It is comparable to the -lh5 switch on the later versions of lharc.

8. Using SETUP.PRG

The first program to be run is SETUP on the installation disk. This will initialise a minixFS partition and create a skeleton directory structure which INSTALL will start to fill later. SETUP has a few options which will be explained below:

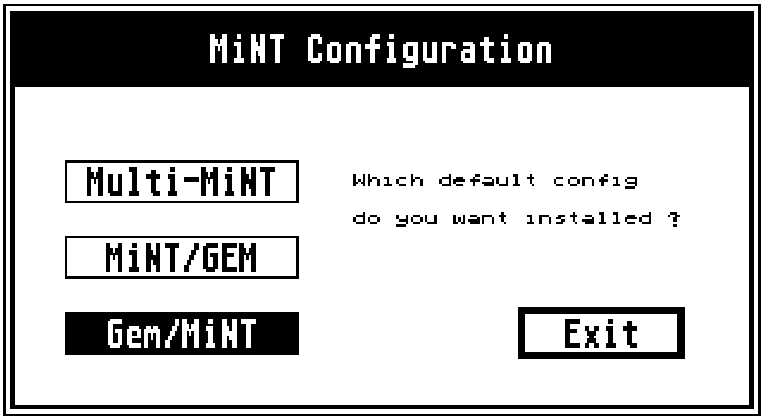
There are 4 menu-bar entries. The first is the familiar 'Desk' entry, the second is the File menu where there are 4 options:

Set root:	Tells SETUP which partition you want to install onto. Make sure you select the root directory of a free partition.
Save Info:	Saves a small configuration file shared by SETUP and INSTALL. This just saves the location of zoo.ttp and the partition to install on.
Install Config:	Fires up the installation process. There is a 'are you sure?' dialogue because this is a dangerous operation.
Quit:	Exits the program.



The 3rd menu is 'Options', which allows you to specify which type of MiNT system you want, and which sort of filesystem you want to run under. The dialogue opposite appears when you click on 'select Filesystem'.

The TOS fs is the normal Atari 8-bit 8+3 filenames etc. If, however, you choose the (default) MinixFS filesystem, you can have up to 32 characters in a filename, along with UPPER and lower case distinction, as many '.'s in a filename as you wish , and the ability to hide the file from casual viewing by making the first character a full stop. The disadvantage of the Minixfs is that MiNT will have to be running for you to be able to access the partition. The system makes sure that TOS will not overwrite the minixFS when MiNT is not active. This protection is only active for HD software released in the last 5 years. Be warned!



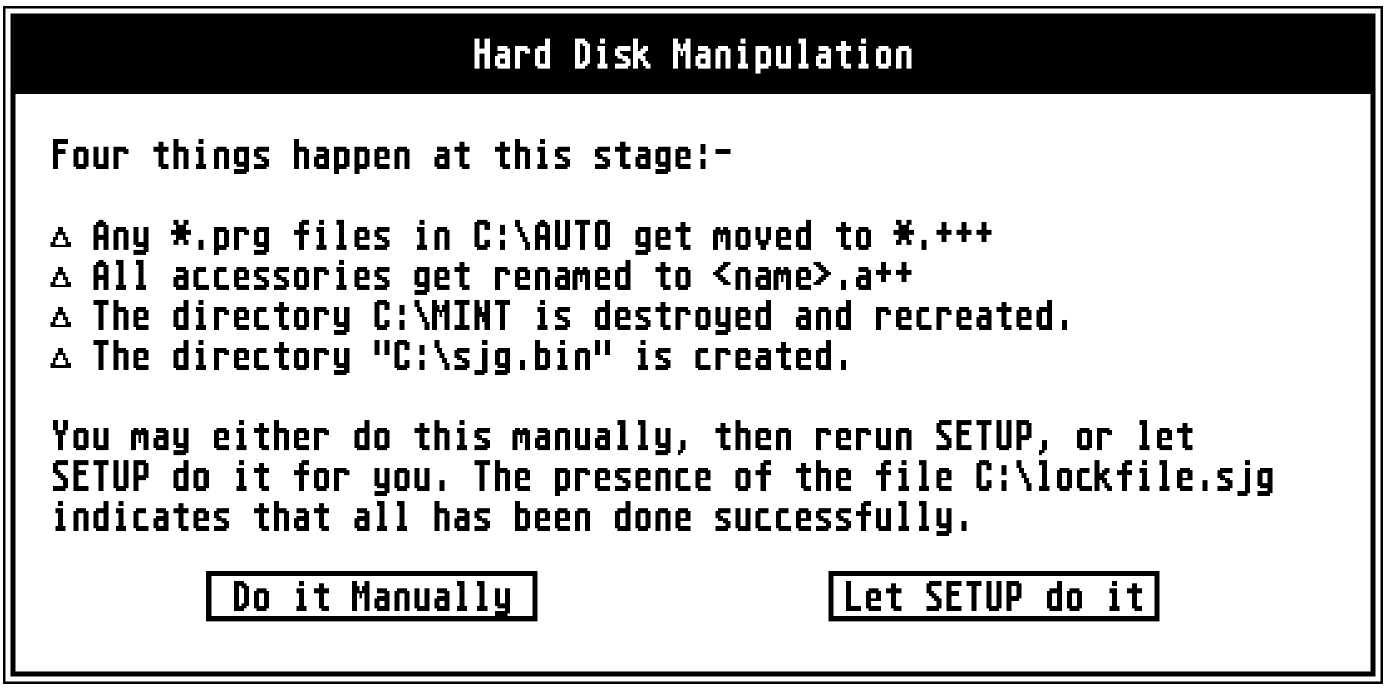
The other choice is which type of MiNT system you want to set up. Actually SETUP will build all 3 configs, so this is no longer as important. As a default it will install the Gem/MiNT configuration (gemmint.cnf in C:/mint).

Gem/Mint allows you to run on the GEM desktop, with TOS programs having their own window, called from the TOSWIN accessory. This is by far the most popular approach. Anyone who has used an xterm will feel at home with TOSWIN.

MiNT/gem (mintgem.cnf) starts you up as root inside a VT52 shell, from where it is to all intents and purposes a Un*x terminal. To run gem from this stage, either type 'gem' (runs gem.ttp), which keeps MiNT active, or press CTRL-D (for end-of-file), which will terminate MiNT and gem will then start up as if MiNT had never happened.

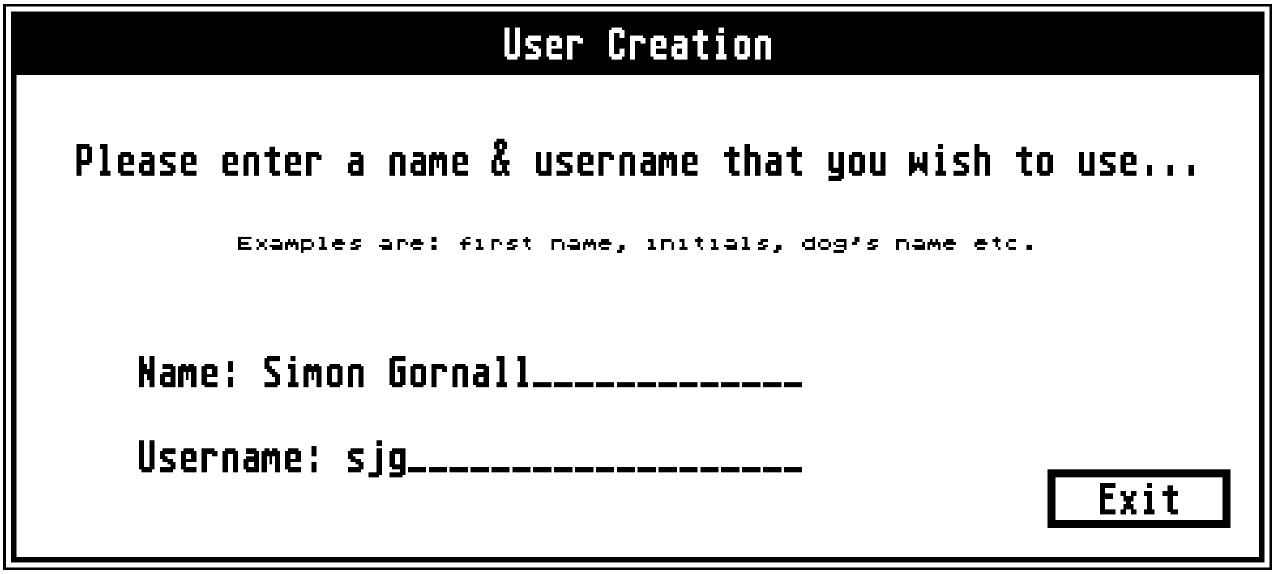
Multi-MiNT (multimnt.cnf) is the multi-user configuration. For this you must have the init subset installed. I assume that anyone preparing to make their ST into a multi-user machine knows enough about Un*x to not need any hand-holding :-) Sorry, but this isn't the place for a complete Un*x tutorial...

After all that, a click on 'Install Config' will start the whole process off. The program tells you what it is about to do for all the important bits. The first major dialogue is when SETUP reports that it is about to change the filenames of accessories and auto-folder programs on your C:\ partition. The reason is that there are some accessories and programs that interfere with MiNT. The program will allow you to quit if you wish, (eg: if you already have a directory called C:\mint). The dialogue looks like that below:



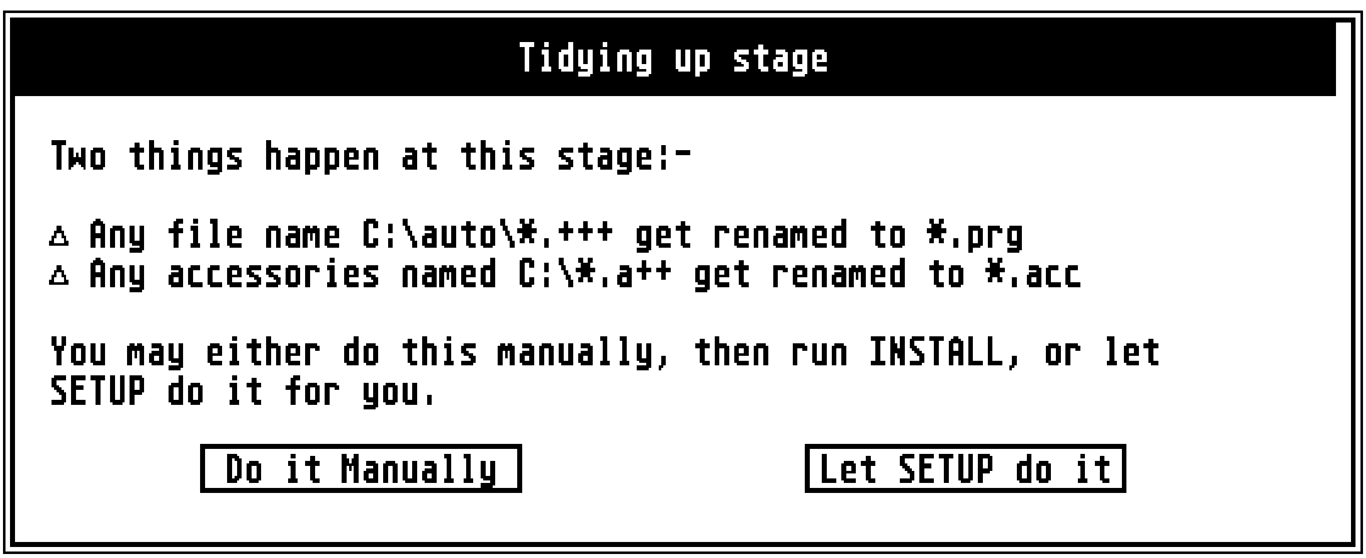
If you now let SETUP do its stuff, it will copy files from A:\ and will create the three config files alluded to above (multimnt.cnf, gemmint.cnf, and mintgem.cnf).

Setup will also ask you for a username and name so it can enter you into the password file. This is so that on a minixFS system the command ‘ls -l’ will report the correct ownerships of files. It is really only of use to multi-MiNT because you are considered to be root otherwise (uid and gid =0). When you get asked for name / username replace mine with ones of your own



The next major thing is to reset the ST(e) so that MiNT can be booted, and the installation restarted by you running SETUP again, and re-clicking on ‘Install Config’. The next stage (this is optional, but highly recommended) is the installation of the minixFS onto the partition. You will then have to reset the ST(e) once again, so that the HD software can recognise the changed type of partition. I know for a fact that ICD 5.5.0 does **not** deal with changed partition type gracefully. AHDI I am unsure of. I expect it comes as quite a shock for the software to find a completely different type of partition halfway through a session. So, reset, run SETUP and click on ‘Install Config’ for the last time. A skeleton directory structure and some files will be copied onto your new partition. Then SETUP will report that it is protecting the files on your filesystem. On a minixFS partition this is important. On a TOS fs partition it will quietly fail and no harm will be done.

Once that is done, you will have to reset the ST(e) for (hopefully) the last time, so that the symbolic links will be installed properly. A dialogue box will appear first ...

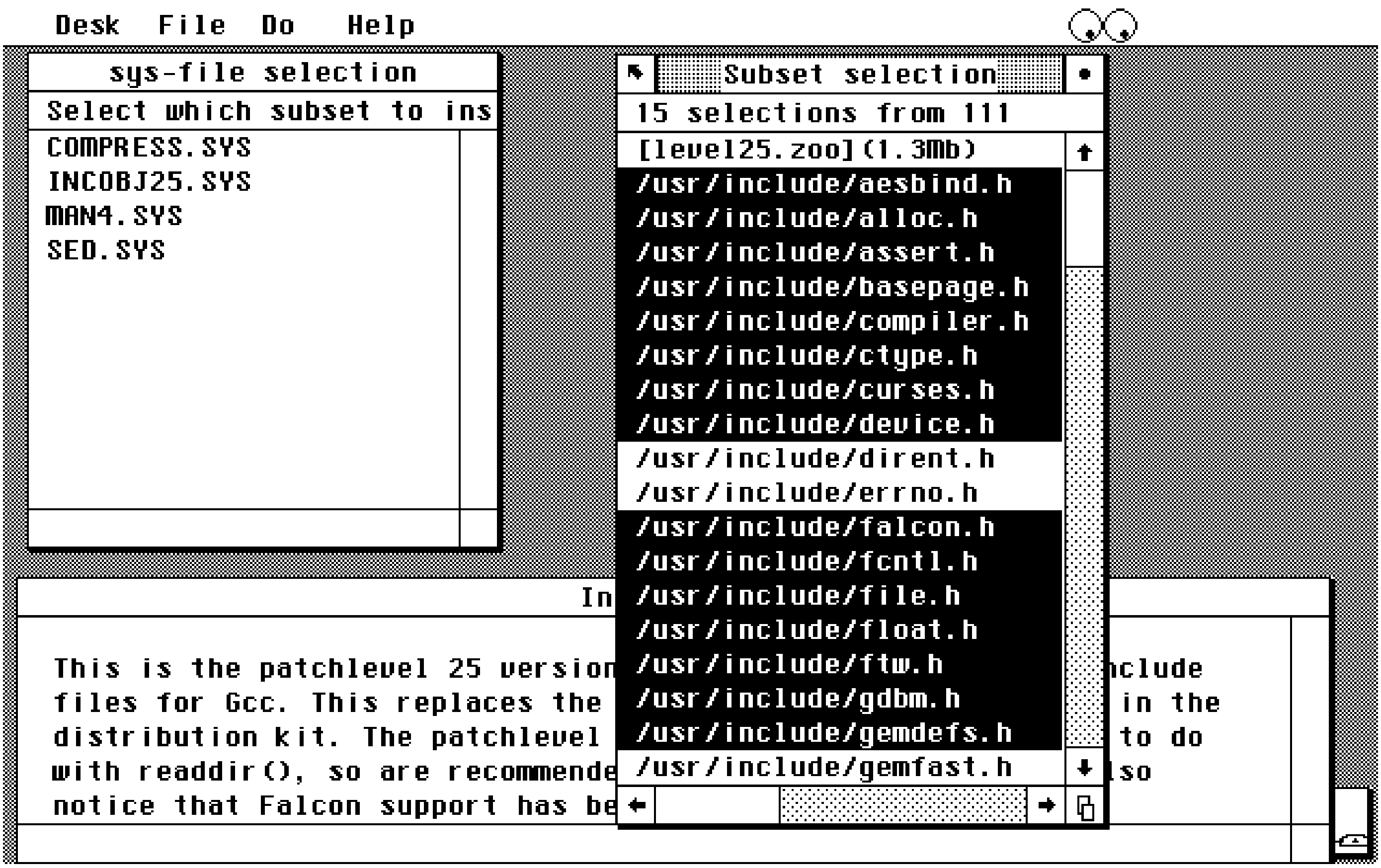


Now you ought to be ready to run INSTALL and finally get the MDK onto your hard disk.

9. Using INSTALL.PRG

INSTALL is a general purpose installation utility that unpacks onto a given partition from .zoo archives, using a configuration file named <something>.sys, and which will recognise bourne shell scripts as files named *.pre and *.pst. It will pipe these scripts through to a bourne shell, that is assumed to be residing in /bin on the installation partition. Appendix A details the configuration file.

In general, the desktop when you're running INSTALL will look something like:



... Or at least, it ought to. The File menu has a couple of options similar to SETUP, but it also has an option 'Load Config', which will load in the configuration file <anything>.sys on the floppy disk (A: or B:). At this point the two windows 'Sys-file selection' and 'Subset selection' ought to pop up. The subset-selection window contains a list of the relevant .sys files on the floppy. Clicking on one of those will put the list of files that the .sys file knows about into the subset-selection window. Clicking on lines in the subset-selection window will toggle them individually for installation or not. They start off as not installed. The name of the subset will appear at the in square brackets, next to an estimate of the size of the files once unpacked from their archive. If you click on a subset name it (de)selects all the elements of the subset.

To actually do the installation, select 'Install configuration' from the *Do* menu. If the 'Run Script' entry is selectable (not greyed out) then there is a script that must be run on the floppy disk. Selecting this will bring up a file selector box, listing all the scripts on the floppy. If the script name is *.pre then it must be run <u>before</u> the subset is installed. If it is a *.pst script it must be run <u>after</u> the subset has been installed. For example the script init.pst must be run after the subset init has been installed.

Warning: Install.prg knows nothing of what you are installing, and the MDK 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 around. Be very careful not to cripple INSTALL because the programs you have INSTALLed haven't the functionality of the originals. Be very careful of programs that go in /bin ! However, they are all on disk1.zoo so you can just recopy them if you ever need to.

10. In use... and the Un*x Credo.

Once you have installed all that you want to put onto your HD, you can reboot and MiNT ought to start up with TOSWIN starting as an accessory, and the GEM desktop starting up as normal. At this stage you will probably want to edit some of the configuration files for the shell that you have chosen to use.

You will find that sh.ttp (and its clones - bash, ash, etc.) look in a file called /etc/profile - at least they're supposed to. I've found that sh doesn't seem to realise it's a login shell, and doesn't read /etc/profile, in which case you can type '. /etc/profile', or put a file '.profile' in your home directory.

(T)csh uses different files - /etc/login.csh is the equivalent in /etc, and tcsh.rc is the relevant file in your home directory.

I've made SETUP generate a (very) bare set of the above files to get you started. You'll probably want to change them from my personal settings. If you have problems getting the shells to read the files in your home directory, put a line:

setenv HOME /usr/users/<username>

... in C:\mint.cnf, which ought to do the trick. Another possibility (especially if you're a user of gemini) is to use GEMterm instead of TOSWIN. GEMterm allows you to specify a home directory from within it's dialogue box so you won't have any problems. In many ways it's nicer than TOSWIN.

For those not using TOSWIN or GEMterm, it is slightly more difficult. Under multi-MiNT you need to change your password entry in /etc/passwd to change your shell or login dir. Under mint-gem, you need to alter the C:\mintgem.cnf file to say INIT=/bin/tcsh.ttp (for example) rather than /bin/sh.ttp to change your shell, and there is a 'cd U:/bin' command which decides where you will start from. Change that and you change your home directory. Remember to keep the U: as a drive specifier though. This is the unified filesystem, all your drives are available as subdirs of U:, for example U:/f is the same as F:/. The point is that you can 'mount' /usr on a different disk drive if you feel the need to expand the system.

Un*x is an ideal as much as it is an operating system. Everything is a file, (screen, keyboard, files, devices, etc.), but also it supplies a vast array of tools to do almost any programming job. These tools are all (in general) small neat utilities that do a single job, but do it well, with switches to specify how they ought to modify their behaviour. The combination of treating everything as a file and having a huge number of versatile excellent utilities to do specific jobs <u>that can act on any file</u> creates a very powerful environment. Add in the ability to pipe the output of one program into the input of another, and you end up with an almost ideal system that is very 'clean'. An example line from my .cshrc in work is:

alias mine 'ps -aux | grep sjg | more'
or
'get all processes in a list' | 'select only those containing my username' | 'page them'

... and set that to be a new pseudo-command called 'mine'. (I have a similar command 'others'). And then of course there are the shell languages - highly programmable in their own right, coupled with things like sed, lex, perl, yacc, and millions of others. This is the environment I would bring to the ST/TT/Falcon, and the future generation of Atari computers. And hence the MDK. I want to thank the authors of the many programs I have incorporated in this kit, not least of all Eric Smith - the effort expended by the few is phenomenal, but the rewards are enjoyed by the many.

If you want to get in touch with me I will (for at least the next 18 months) be contactable as sjg@phlim.ph.kcl.ac.uk (Internet) or cbs%uk.ac.kcl.ph.ipg::sjg (JANet). I'd like to hear from anyone with problems, suggestions (clean, please), or comments. Have fun!

Simon.

Appendix A: The configuration file format.

The <anything>.sys file that INSTALL uses to parse the .zoo files consists of the following:

A line:

    [name of zoo file](size when unpacked)   owner-id group-id permissions

... at the head of the file, where owner, group, and permissions are the default for the entire subset. There must be spaces between the ')' and the owner-id. Obviously there must also be spaces between the owner-id, group-id, and permissions fields.

Any number of lines:

```
# This is a comment. INSTALL will place these lines in the Info... window
# when the subset has been selected. Place a space after the # and then keep
# to less than 80 chars for best effect.
```

A list of lines:

    /xxx/yyy/filename   owner-id group-id permissions

... where the owner-id, group-id, and permissions are optional. If present they override the defaults for this subset, as set at the head of the file. The permissions field is set in traditional Un*x style, as a 3 or 4 digit octal number. If any of the fields are present then they all must be. Also note that spaces, not tabs are necessary between the filename and the fields if present. Note that the filename field must be exactly what is stored in the .zoo file, and will be exactly what is placed on the disk, with directories created if need be.

An example follows of a short .sys file:

```
[comprss.zoo](37k) 0 0 755
#
# Compress is the standard un*x compression utility. Actually there is
# 'compact' on unices as well, but compress seems to be used *much* more
# frequently. This is a TOS-friendly compress routine, ie: it does not
# append a ".Z" to the filename, but appends "z" instead, and truncates to
# 8+3 filename format.
#
/usr/bin/compress.ttp
/usr/man/man1/compress.1   0 0 644
```