#ident "@(#)proto:i386/README 1.1.1.1"

1. Building Binary from Source[1]

This section describes the procedure for building binaries from the source tape. The C Software Development Set Issue 4.1.5 source is part of the source tape. The system cannot be built with an earlier release of the C Software Development Set and cannot be built with the UNIX® System V Release 3.1 C Software Development Set.

The international version of the Source Code Provision differs from the domestic version in that the following two files are *NOT* included in the international version:

— $ROOT/usr/src/cmd/crypt/crypt.c

— $ROOT/usr/src/lib/libcrypt/des_decrypt.c

1.1 Source Installation

The source tape contains the usr/src directory. All source code (and some object files) are under this.

To install the source code from the distributed cartridge tape, perform the following steps:

a. Make a root directory, /usr/src386[2], for the source tree.
```
mkdir /usr/src386
```

b. Insert the source tape and retention the source tape.
```
/usr/lib/tape/tapecntl -t
```

c. Change into this directory and read in the source tape.
```
cd /usr/src386
cpio -icBdum < /dev/rmt/c0s0
```

The following binaries are included the Source Code Provision:

— $ROOT/usr/src/uts/i386/io/ansi.o *

— $ROOT/usr/src/uts/i386/io/asy.o

— $ROOT/usr/src/uts/i386/io/cpyrt.o

— $ROOT/usr/src/uts/i386/io/cram.o

— $ROOT/usr/src/uts/i386/io/fd.o

— $ROOT/usr/src/uts/i386/io/hd.o

— $ROOT/usr/src/uts/i386/io/kd.o *

— $ROOT/usr/src/uts/i386/io/kdkbtables.o *

— $ROOT/usr/src/uts/i386/io/lp.o

— $ROOT/usr/src/uts/i386/io/rtc.o

— $ROOT/usr/src/uts/i386/io/vtgen.o *

— $ROOT/usr/src/uts/i386/io/xque.o *

---

1. The building of the binary can be done either as root or non root.

2. In this document, /usr/src386 is used as an example for the 'root' directory. You can use any directory you wish for your 'root' except for /. If you use a root other than /usr/src386, please remember your root as you read this document.

* The $ROOT/etc/conf/pack.d/kd/Driver.o consists of the following modules from $ROOT/usr/src/uts/i386/io: ansi.o, kd.o, kdkbtables.o, vtgen.o, xque.o

— $ROOT/usr/src/uts/i386/fp/emulator.ms

1.2 Building the Release

This section describes a way of building the binary release for the UNIX® System V/386 Release 3.2[3]. To build a binary release including kernel, commands, libraries, C Software Development Set and other add-on packages, perform the following steps:

a. Set and export the environment variable $ROOT to the root of the source tree.
```
ROOT=/usr/src386; export ROOT
```
[4]

b. Change into the directory containing the build scripts and build the code.
```
cd $ROOT/usr/src
nohup sh ./:mk &
```
This will build and install the complete UNIX® System V/386 Release 3.2 relative to $ROOT. The new binaries will not interfere with the running system. This step takes about 24 hours on a 386 system with 2 megabytes of memory. We *nohup* the build in background so we can look at the build output later if needed.

1.3 Build only the Foundation Set and Extended Terminal Interface Package

To Build only the Foundation Set and the Extended Terminal Interface Package and not the C Software Development Set, perform the following steps:

a. Set and export the environment variable $ROOT to the root of the source tree.
```
ROOT=/usr/src386; export ROOT
```

b. Change into the directory containing the build scripts and build the code.
```
cd $ROOT/usr/src
nohup sh ./:mk.fnd &
```
This will build and install the Foundation Set and the Extended Terminal Interface Package relative to $ROOT. The new binaries will not interfere with the running system. This step takes about 8 hours on a 386 system with 2 megabytes of memory. We *nohup* the build in background so we can look at the build output later if needed.

1.4 Build only the C Software Development Set

To Build only the C Software Development Set and not the Foundation Set nor the Extended Terminal Interface Package, perform the following steps:

a. Install the Extended Terminal Interface Package Version 3.0. This is needed in order to compile the cscope(1) command.

b. Set and export the environment variable $ROOT to the root of the source tree.
```
ROOT=/usr/src386; export ROOT
```

c. Change into the directory containing the build scripts and build the code.
```
cd $ROOT/usr/src
nohup sh ./:mk.csds &
```
This will build and install the C Software Development Set relative to $ROOT. The new binaries will not interfere with the running system. This step takes about 11 hours on a 386 system with 2 megabytes of memory. We *nohup* the build in background so we can look at the build output later if needed.

---

3. Only the AT386 bus type is supported by default - without modification

4. Where /usr/src386 is an example of a user defined directory containing a source tree and where executables will be placed.

1.5 Build only the kdb, rfs, nsu, s52k, xx and termpkg Add-ons

To Build only the Kernel DeBugger (kdb), Remote File Sharing Package (rfs), Network Support Utilities Package (nsu), 2 Kilobyte File System Utility Package (s52k), XENIX™ File System Package (xx) and the Remote Terminal Package (termpkg), perform the following steps[5]:

a. Set and export the environment variable $ROOT to the root of the source tree.
```
ROOT=/usr/src386; export ROOT
```

b. Set up your PATH to include the $ROOT/xenv
```
PATH=$ROOT/xenv:$PATH:/etc; export PATH
```

c. Change into the directory containing the build scripts and build the code.
```
cd $ROOT/usr/src
sh ./.setup[6]
MCS=mcs; export MCS
nohup sh ./:mk.addon kdb rfs nsu s52k xx &[7]
```

This step takes about 2 hours on a 386 system with 2 megabytes of memory. We *nohup* the build in background so we can look at the build output later if needed.

This will build each of these add-ons and create a floppy image of each under /tmp. See Section 7 item l for more details.

d. Format a 1.2 Meg Floppy diskette.
```
format -i2 /dev/rdsk/f0q15dt
```

e. Build the Remote Terminal Package and follow the prompts on the screen.
```
sh ./:mk.addon termpkg
```

1.6 Build only UNIX® Operating System Kernel

To Build only the kernel and nothing else, perform the following steps[8]:

a. Set and export the environment variable $ROOT to the root of the source tree.
```
ROOT=/usr/src386; export ROOT
```

b. Set up your PATH to include the $ROOT/xenv
```
PATH=$ROOT/xenv:$PATH:/etc; export PATH
```

c. Change into the directory containing the build scripts and build the kernel.
```
cd $ROOT/usr/src
sh ./.setup
nohup sh ./:mkuts AT386 &
```

This will build and install the kernel relative to $ROOT. This step takes about 1 hour on a 386

---

5. Note: The build of kdb, rfs, nsu, s52k and xx depends on the following libraries: libns.a and libnsl_s.a. If you have not already built the Foundation Set, you must build these libraries *first* before you build these packages or else the build *will* fail.

   To build these libraries, perform this step before step 'c':
   ```
   cd $ROOT/usr/src
   CH=#; export CH
   sh ./:mklib libns nsl
   ```

6. This step sets up the front end scripts to make(1) and cc(1). Setup also runs :mkhead and :mksyshead if needed.

7. The Remote Terminal Package cannot be built in background.

8. Note: The build of the kernel depends on the Installable Driver Commands being located relative to $ROOT. Therefore, if you have *not* already built the Foundation Set, you must perform the following steps:
   ```
   mkdir $ROOT/etc
   cd /etc
   find conf/bin -print | cpio -pdum $ROOT/etc
   ```

system with 2 megabytes of memory. We *nohup* the build in background so we can look at the build output later if needed.

1.7 Building Selected Parts of the Source Code Provision

The following commands build individual parts of the system[9].

./:mklib libcurses	Builds and installs the curses and termcap library.
./:mklib lxcurses libm	Builds and installs the XENIX™ version of curses and the math libraries.
./:mklib *	Builds and installs all the libraries found under usr/src/lib
./:mkcmd *	Builds and installs all the commands found under usr/src/cmd (except .adm).
./:mkcmd who	Builds the who command and installs it relative to $ROOT.
./:mk.addon *	Builds all the add-ons found under usr/src/add-on. Makes floppy images of each under /tmp. Prompts the user for the termpkg floppy.
./:mk.addon rfs	Builds the Remote File Sharing Package Add-on and makes a floppy image of the installable floppy under /tmp/PID.rfs.

2. Packaging the binary output

To package the newly created binary product, the following steps must be performed:

a. Use the procedure outlined above, *6.2 Building the Release.*

b. Change into $ROOT/usr/src/proto/i386/at386/cmd
```
cd $ROOT/usr/src/proto/i386/at386/cmd
```

c. Become root and set your PATH.
```
/bin/su root
PATH=$ROOT/xenv:$PATH; export PATH
```

d. Compile the setmods program.
```
make -f cmd.mk ../setmods
```

e. Prepare the C Software Development Set package[10]:
```
cd $ROOT/usr/src/proto/i386/at386/PACKAGES/csds
echo bin/make | sh ../../../prep NO_UNIX
```

---

9. Note: Before you run these commands, you must build the front end scripts to make(1) and cc(1). To do this type in:
./.setup.
You must also add $ROOT/xenv and /etc to your PATH. To do this, type in:
PATH=$ROOT/xenv:$PATH:/etc; export PATH

```
make -f csds.mk prep
```

f. Package the Base System Package

```
cd $ROOT/usr/src/proto/i386/at386
make -f proto.mk package
```

g. Format 12 Double Sided, High Density floppy diskettes.

```
format -i2 /dev/rdsk/f0q15dt
```

h. Package the following packages:

— Editing Package

— Security Administration Package [11]

— Extended Terminal Interface Package

```
cd $ROOT/usr/src/proto/i386/at386/PACKAGES
make -f PACKAGES.mk install clobber [12]
```

i. Package the C Software Development Set package:

```
cd $ROOT/usr/src/proto/i386/at386/PACKAGES/csds
make -f csds.mk install clobber
```

j. Return from super-user shell *(that was created in step 7c).*

```
exit
```

k. Build and package the Remote Terminal Package (termpkg)

```
cd $ROOT/usr/src
PATH=$PATH:/etc; export PATH
sh ./:mk.addon termpkg
```

l. Package the following packages:

— Kernel DeBugger (kdb)

— Network Support Utilities Package (nsu)

— Remote File Sharing Package (rfs)

— Two Kilobyte File System Package (s52k)

— XENIX™ File System Package (xx)

a. Determine the files to be put on to the diskettes.

```
cd /tmp
ls -l *kdb *nsu *rfs *s52k *xx
```

---

10. If the C Software Development Set was not built, they you can skip this step. However, if the C Software Development Set was built, then this step must be performed before 'Package the Foundation Set'. This is because some of the Installable Driver Commands are really links to the C Software Development Set counter parts.

11. For domestic customers only.

12. For the international customer, the packaging of the Security Adminstration Package and the Extended Terminal Interface Package will fail. Therefore, in order for the international customer to package the Extended Terminal Interface Package, the following steps must be done.

    — cd $ROOT/usr/src/proto/i386/at386/PACKAGES/GPU

    — make -f GPU.mk install clobber

    — cd $ROOT/usr/src/proto/i386/at386/PACKAGES

    — make -f PACKAGES.mk clobber

b. For each of these files, dd it on to a diskette.

```
dd if=/tmp/NAME of=/dev/rdsk/f0q15d bs=15b
```

where NAME is the name of the cpio archive for the given package.

3. Freeing Disk Space

Several groups of files may be removed after you have built the system if disk space is needed. These groups are located in several areas. To remove them change into the $ROOT directory and then perform the following steps:

3.1 To clean up space in the kernel build tree

a. cd $ROOT/usr/src/uts/i386

b. make -f unix.mk clean

3.2 To clean up space in the shared library build tree

a. cd $ROOT/usr/src/lib/shlibc

b. rm host

3.3 To clean up space in the proto build tree

a. cd $ROOT/usr/src/proto/i386/at386

b. make -f proto.mk clobber

3.4 To clean up space in /tmp (the add-on floppy images)

a. cd /tmp

b. rm -f *kdb *nsu *rfs *s52k *xx

4. Warning Messages

Warning messages will occur when you build the Source Code Provision. The errors that issue these warning will *not affect* the build and should be ignored. These messages are:

Message	When Building
line NN: warning: illegal pointer/integer combination, op =	lib/libc lib/libPW lib/shlib lib/lxcurses cmd/cscope cmd/sgs/mkshlib cmd/sgs/strip uts/i386/io/clist.c
line NN: warning: illegal pointer/integer combination, op ==	lib/libc lib/shlibc
line NN: warning: loop not entered at top	cmd/xrestore
warning(NN): template yyy can run out of registers	cmd/sgs/comp2
*** Error code 2 (ignored)	cmd/login cmd/sccs cmd/spell cmd/tar cmd/xrestore uts/i386/boot
line NN: warning: illegal member use: u_fpstate	cmd/sdb
Extra token (ignored) after directive	cmd/ls
mkdir: Failed to make directory "XXXXXX"; File exists	cmd/login cmd/sccs cmd/spell cmd/tar cmd/xrestore uts/i386/boot
line NN hides line yy	cmd/sgs/comp2
install: -u option available only to root -- ignored	
install: -g option available only to root -- ignored	
strip: Warning - symbol directory deleted from archive libPW.a	lib/libPW
mcs: Warning archive symbol table deleted from ’NN’	add-on/nsu Packaging the Base System Packaging the Add-ons
ssignal.c: 16: MAXSIG redefined	lib/libc lib/shlibc
param.h: 49: NOFILE redefined	cmd/csh
lex.c: 14: ECHO redefined	cmd/shl
**** Build of YYY failed (no makefile found in directory)	cmd/include cmd/lib
llib-lc(NNN): syntax error	cmd/lint
tic: Warning: near line 37 (or 39): terminal ’YYY’, ’YY’ defined in more than one entry.	cmd/lp
Null or unknown MCS variable passed to WWW. Object comment section will remain unchanged.	Running :mk.addon
mcs: Couldn’t rewrite ’bin/make’	Packaging the CSDS
./usr/include/sti.h: No such file or directory	Packaging the Entended Terminal Interface.
**** Build of crypt failed (make)	cmd/crypt for the international customer.
Line 1: bin/crypt does not exist Line 62: usr/lib/libcrypt.a does not exist	Packaging the security administration package for the international customer

Messages that are NOT any of the above must be carefully scrutinized.