Compiling and Booting the Nsys UNIX Kernel

Warren Toomey
Email: wkt@cs.adfa.edu.au

15th January, 1999

Introduction

The 'nsys' UNIX kernel was recently donated to the PUPS Archive by Dennis Ritchie. This file describes how you can boot a slightly-modified version of this kernel on a 5th Edition RK05 root filesystem.

Background

In January 1999, Dennis Ritchie sent in a copy of the 'nsys' UNIX kernel for inclusion in the PUPS Archive. In the accompanying README, he says:

> So far as I can determine, this is the earliest version of Unix that currently exists in machine-readable form. ... The dates on the transcription are hard to interpret correctly; if my program that interprets the image are correct, the files were last touched on 22 Jan, 1973. ...
>
> What is here is just the source of the OS itself, written in the pre-K&R dialect of C. The file `u` is evidently a binary image of the kernel. It is intended only for PDP-11/45, and has setup and memory-handling code that will not work on other models (it's missing things special to the later, smaller models, and the larger physical address space of the still later 11/70.) It appears that it is intended to be loaded into memory at physical 0, and transferred to at location 0.
>
> I'm not sure how much work it would take to get this system to boot. Even compiling it might be a bit of a challenge, though I imagine that circa V5-V6 compilers would handle the language (maybe even V7). It is full of `=+` and use of `char *` as unsigned integers, and integers used as pointers in locations like `p->x`.
>
> So far as I can determine, the disk format it expects is compatible with the layout of other earlyish research systems (V5, V6) for example. But perhaps not, and it's not certain that the source is complete. Even the compilation mechanism is a bit unclear, though it certainly used the shell script `ken/rc`, which appears to depend on having the *.o files from `ken/*` and `dmr/*` and also `slib.a`, which doesn't exist on the tape. My guess is that this was an archive of already compiled .o files, so that (for example) to test-build a system one would edit a file, compile it, and run `ken/rc` to load it. The 'ustr' routine referred to in ken/rc evidently stripped off the a.out header and the symbols from the a.out file.
>
> Best wishes with this. I'd be interested to hear whether anyone gets the `u` image to run. If you're in luck, all you need is an 11/45 processor or emulator and a V5/6/7 disk image.

I decided to try and compile the 'nsys' code, and get it to boot on a PDP-11/45 emulator. A secondary goal was to use the 'nsys' compilation effort to find any remaining bugs in my Apout PDP-11 user-mode emulator.

Tools Required

Because the 'nsys' kernel is dated 1973, I decided to use a 5th Edition development enviromnent to compile it. This minimised the changes I had to make to the source code.

There are several ways you can obtain a 5th Edition development enviromnent:

- Load a 5th Edition filesystem on an RK05, and boot it on a PDP 11/45;
- Load a 5th Edition filesystem on an RK05 disk image, and boot it on a PDP 11/45 emulator; or
- Unpack the 5th Edition root filesystem tree on a 32-bit little-endian Unix system, and use the Apout emulator.

For convenience, I chose to use the Apout emulator (version 2.2alpha8) as my development environment, and Bob Supnik's PDP-11 simulator (version 2.3d) as the booting environment. With the latter, I used the 5th Edition RK05 disk image *Distributions/research/Dennis_v5/v5root.gz* (from the PUPS Archive) as the filesystem.

Regardless of the method you choose, the modified 'nsys' source must be unpacked in the directory `/sys/nsys`, relative to the top of the 5th Edition root directory, i.e the file `u` (size 26,266 bytes) becomes `/sys/nsys/u`. This file is the original 'nsys' kernel image, as supplied by Dennis; we will build our own kernel image.

Changes to the Nsys Source

Several files in the 'nsys' source had to be modified so as the 'nsys' kernel would work with a 5th Edition filesystem & boot environment. The discovery of these changes took me several days of fiddling with C code, assembly code, single-stepping machine code, and perusing the Lions' commentary. Email me if you really want to know the gory details.

The changes to the ten 'nsys' source code files are described below:

Filesystem The 5th Edition filesystem is laid out slightly differently to that which the 'nsys' kernel is expecting. The `filsys` struct needs an extra field, `s_ronly`, and the `inode` struct needs an extra field, `i_lastr`. The two files affected are `filsys.h` and `inode.h`.

C language The C language changed slightly from the 'nsys' kernel to the 5th Edition. Sub-structures defined within a structure were delimited by parentheses in 'nsys', but by braces in 5th Edition. The only file affected is `user.h`.

Several lines in the `psig()` routine in `ken/sig.c` were rearranged because the 5th Edition C compiler refused to parse them. The actual operations performed are unchanged.

Device table changes The 'nsys' kernel is configured to have four block device drivers (rf, rk, tc and tm), and ten character device drivers (kl, dc, pc, dp, dn, mm, vt, da, ct, vs). To minimise debugging, I chose to remove as many of the drivers as possible. I left two block device drivers, rk and tm, and two character device drivers, kl and mm. The two main files affected were `conf/c.c` and `conf/l.s`.

The device driver `dmr/rk.c` also had to be modified, as it was hard-wired to be block device number 1. It is now block device number 0.

Deficiencies In the `putchar()` routine in `prf.c`, a test is made on a register in the console KL device. This register isn't described in my PDP-11 peripherals handbook (dated 1973), and it isn't implemented in Bob Supnik's simulator, so I removed the code.

The declaration of the `clist` struct in `tty.h` need a semicolon to end the declaration; again, this could be due to a change in the C language.

The machine code for location 0 in `conf/l.s` has the octal value 4 then the instruction `br 1f`. In 5th Edition, these two are transposed. It appears that the entry to the 'nsys' kernel must have been at

location 2 (i.e the `br` instruction), whereas the 5th Edition kernel starts at location 0. Octal 4 is an IOT instruction, which causes an immediate hardware exception on the PDP-11. I have transposed these two lines in the 'nsys' code.

Cosmetic changes While trying to get the 'nsys' kernel to boot properly, I added the 5th Edition `printf` line to `main()` in `main.c`, which outputs the amount of physical memory available on the machine.

As well as these changes, I have moved a few files around so that those parts of the kernel which must be tailored for each hardware configuration are kept in the `conf/` directory. Final linking of the 'nsys' kernel is also done in this directory. A few other files have been renamed or moved, again to tidy up the layout of the source. The changes are:

- `dmr/malloc.c` becomes `dmr/mem.c`
- `tables.c` becomes `dmr/partab.c`
- `ken/45.s` becomes `conf/mch.s`
- `ken/low.s` becomes `conf/l.s`
- `ken/conf.c` becomes `conf/c.c`

There are several new RCS directories, which hold the changes to the ten 'nsys' files listed above. Finally, three short shell scripts have been created to make compilation of the 'nsys' kernel relatively easy: `ken/mklib`, `dmr/mklib` and `conf/mkunix`.

Compiling the Nsys Kernel

Here is a typescript of the commands required to compile the 'nsys' kernel. I have added some comments to the typescript.

```
% alias 5sh
setenv APOUT_ROOT /usr/local/src/V5; apout $APOUT_ROOT/bin/sh

% 5sh                          # Run Apout

# chdir /sys/nsys              Move to the 'nsys' directory
# chdir ken                    Start with the kernel code
# cat mklib
cc -c -O *.c
rm ../lib1
ar vr ../lib1 main.o alloc.o iget.o prf.o rdwri.o slp.o subr.o text.o trap.o \
 sig.o sysent.o clock.o fio.o malloc.o nami.o prproc.o sys1.o sys2.o \
 sys3.o sys4.o

# sh mklib                     Run the script to build lib1
alloc.c:
clock.c:
fio.c:
iget.c:
main.c:
malloc.c:
nami.c:
prf.c:
prproc.c:
rdwri.c:
```

```
sig.c:
slp.c:
subr.c:
sys1.c:
sys2.c:
sys3.c:
sys4.c:
sysent.c:
text.c:
trap.c:
../lib1: non existent
a main.o
a alloc.o
a iget.o
a prf.o
a rdwri.o
a slp.o
a subr.o
a text.o
a trap.o
a sig.o
a sysent.o
a clock.o
a fio.o
a malloc.o
a nami.o
a prproc.o
a sys1.o
a sys2.o
a sys3.o
a sys4.o

# chdir ../dmr                          Move to the devices directory
# cat mklib
cc -c -O *.c
as gput.s
mv a.out gput.o
rm ../lib2
ar vr ../lib2 *.o

# sh mklib                              Run the script to build lib2
bio.c:
cat.c:
dc.c:
dn.c:
dp.c:
draa.c:
kl.c:
mem.c:
partab.c:
pc.c:
rf.c:
rk.c:
tc.c:
tm.c:
tty.c:
```

```
vs.c:
vt.c:
../lib2: non existent
a bio.o
a cat.o
a dc.o
a dn.o
a dp.o
a draa.o
a gput.o
a kl.o
a mem.o
a partab.o
a pc.o
a rf.o
a rk.o
a tc.o
a tm.o
a tty.o
a vs.o
a vt.o

# chdir ../conf                          Move to the configuration directory
# cat mkunix
as mch.s
mv a.out mch.o
cc -c c.c
as l.s
mv a.out l.o
ld -x l.o mch.o c.o ../dmr/gput.o ../lib1 ../lib2
mv a.out unix
nm -n unix > namelist
ls -l unix
size unix

# sh mkunix                              Build config, link the kernel
-rwxrwxrwx  1 root    25322 Jan 14 22:02 unix
21286+888+15962=38136 (0112370)

# ls -l                                  And see what other files we have
total 82
drwxr-xr-x  2 root      512 Jan 14 05:37 RCS
-r--r--r--  1 root      307 Jan 14 00:31 c.c
-rw-rw-rw-  1 root      292 Jan 14 22:02 c.o
-rw-rw-rw-  1 root     1200 Jan 14 22:02 l.o
-r--r--r--  1 root     2004 Jan 13 22:37 l.s
-rw-rw-rw-  1 root     1888 Jan 14 22:02 mch.o
-r--r--r--  1 root     3896 Jan 10 18:19 mch.s
-r--r--r--  1 root      161 Jan 14 19:37 mkunix
-rw-------  1 root     3995 Jan 14 22:02 namelist
-rwxrwxrwx  1 root    25322 Jan 14 22:02 unix
#
```

Installing the Nsys Kernel

Now that the 'nsys' kernel is compiled, we have to install it in the root directory of an RK05 5th Edition UNIX root filesystem. I used the bootable 5th Edition disk image `v5root` and Bob Supnik's emulator to do this.

5th Edition UNIX doesn't have *tar*, so I mounted `v5root` as RK pack 0, and I mounted the new 'nsys' kernel as RK pack 1, after doing some padding to the file.

```
% ls -l
-rwx------  1 wkt  wheel   117728 Jan 11 14:02 pdp      Supnik simulator
-rw-------  1 wkt  wheel       55 Jan 15 14:12 v5       Config file
-r--------  1 wkt  wheel  2494464 Jan 15 13:35 v5root   V5 filesystem
-rw-------  1 wkt  wheel     4096 Jan 14 12:06 zero     File of zeroes

% cp ../V5/sys/nsys/conf/unix nsys.binary       Copy the kernel here
% cat zero >> nsys.binary                       Pad it with zeroes

% cat v5                                        Here is the config file
set cpu 18b
att rk0 v5root
att rk1 nsys.binary
boot rk

% ./pdp v5                                      Run the simulator

PDP-11 simulator V2.3d
@unix                                           Start V5 UNIX

login: root
# check /dev/rrk0                               Check root filesystem
/dev/rrk0:
spcl       5
files    566
large    126
direc     28
indir    126
used    3790
last    3985
free     128

# ls -l /dev                                    You may need to /etc/mknod
total 0                                         at least /dev/rrk1
cr--r--r--  1 bin     1,  0 Nov 26 18:13 mem
crw-rw-rw-  1 bin     1,  2 Nov 26 18:13 null
crw-rw-rw-  1 root    2,  0 Mar 21 13:53 rrk0
crw-rw-rw-  1 root    2,  1 Mar 21 14:18 rrk1
crw--w--w-  1 root    0,  0 Mar 21 15:26 tty8

# dd if=/dev/rrk1 count=50 of=z                 Load in 'nsys' kernel + pad
50+0 records in
50+0 records out

# dd if=z of=nsys bs=11761 count=2              Trim back to correct size
2+0 records in
2+0 records out
```

```
# rm z                                          Remove temporary files

# ls -l nsys                                    Check correct size
-rw-rw-rw-  1 root    23522 Mar 21 15:29 nsys

# size nsys                                     Verify a.out values, should
21286+888+15962=38136 (0112370)                 be the same as before

# sync                                          Shut down V5 UNIX
# sync
# ^E                                            and exit the simulator
Simulation stopped, PC: 014116 (BNE 14150)
sim> q
Goodbye
```

You now have the 'nsys' kernel stored in the root directory of the 5th Edition root filesystem. You can now boot it and see that it works.

```
% ./pdp v5                                      Run the simulator

PDP-11 simulator V2.3d
@nsys                                           Load the nsys kernel
mem = 64539                                     Printout of avail memory

login: root                                     /etc/init works!

# ls -l                                         So does /bin/ls
total 107
drwxr-xr-x  2 bin       944 Nov 26 18:13 bin
drwxr-xr-x  2 bin       112 Mar 21 14:21 dev
drwxr-xr-x  2 bin       240 Mar 21 12:07 etc
drwxr-xr-x  2 bin       224 Nov 26 18:13 lib
drwxr-xr-x  2 bin        32 Nov 26 18:13 mnt
-rw-rw-rw-  1 root    23522 Mar 21 15:29 nsys
drwxrwxrwx  2 bin       128 Mar 21 14:16 tmp
-rwxrwxrwx  1 bin     25802 Mar 21 12:07 unix
drwxr-xr-x 14 bin       224 Nov 26 18:13 usr
# sync
# ^E                                            Exit the simulator
Simulation stopped, PC: 015140 (BLT 15050)
```

Final Notes

The 'nsys' kernel can now boot and run some 5th Edition UNIX a.out binaries. However, 'nsys' is an earlier version than 5th Edition, so there will be some V5 functionality which 'nsys' does not support. In particular, 'nsys' does not have the `pipe()` system call. I have only just got the 'nsys' kernel to boot, so I have not had a chance to sit down and work out exactly what functionality is missing.