Installing (and running) BSD 2.11 on a PDP11

Dr. Bernd Ulmann

December 28, 2011

1 Introduction

For my lectures *Operating Systems Theory/Practice* I wanted to have a historic UNIX system up and running for demonstration purposes. Since I managed to build a Frankenstein-PDP11 made from pieces out of various junk bins, it seemed logical to install a BSD UNIX on this machine, namely BSD 2.11. This document describes the installation procedure (which is outlined in way more detail in the official installation and user's guide [2.11BSD]).

Since a real `ts` tape drive is quite slow and since I already have a `simh` emulator up and running on my Mac, I decided to perform the installation of the BSD distribution kit on the emulator and then transfer the disk image to a real disk connected to the real PDP11.

A wealth of information and help has been obtained from `http://www.retrocmp.com/how-tos/installing-211bsd-unix-on-pdp-1144` which describes the process of getting BSD 2.11 up and running on a PDP11/44.

2 Installation

2.1 Booting from tape

First of all start the emulator (the path name shown is valid only for my local installation):

```
alberich$ ../../simh/pdp11

PDP-11 simulator V3.8-1
sim> set tto 7b
sim> set ts enable
sim> set cpu 3072k
Disabling CR
Disabling RK
Disabling HK
Disabling TM
sim> set rq0 ra92
sim> attach ts media/211bsd.tap
attach rq0 media/ra92.dsk
sim> RQ: creating new file
sim> boot ts

73Boot from ts(0,0,0) at 0172522
```

2.2 Disklabel

Since I am lucky to have SCSI controller for the PDP11 I decided to use a rather large disk and settled for the `RA92` implementation of `simh`. The first thing that is necessary is loading the `disklabel` program from tape and perform the necessary disk partitioning magic:

```
: ts(0,1)
Boot: bootdev=01001 bootcsr=0172522
disklabel
Disk? ra(0,0)
'ra(0,0)' is unlabeled or the label is corrupt.
Proceed? [y/n] y
d(isplay) D(efault) m(odify) w(rite) q(uit)? D
d(isplay) D(efault) m(odify) w(rite) q(uit)? d

type: MSCP
disk: RA92
label: DEFAULT
flags:
bytes/sector: 512
sectors/track: 73
tracks/cylinder: 13
sectors/cylinder: 949
cylinders: 3099
rpm: 3600
drivedata: 0 0 0 0 0

1 partitions:
#        size   offset    fstype   [fsize bsize]
  a: 2940951 0  2.11BSD    1024 1024    # (Cyl. 0 - 3098)
```

Now we have a `RA92` disk with one partition named `a` that spans over the whole disk. This disk now needs a label – it will be named `SNOOPY` since that is the PDP11's name:

```
d(isplay) D(efault) m(odify) w(rite) q(uit)? m
modify
d(isplay) g(eometry) m(isc) p(artitions) q(uit)? m
modify misc
d(isplay) t(ype) n(ame) l(abel) f(lags) r(pm) D(rivedata) q(uit)? l
label [DEFAULT]: SNOOPY
modify misc
d(isplay) t(ype) n(ame) l(abel) f(lags) r(pm) D(rivedata) q(uit)? q
modify
d(isplay) g(eometry) m(isc) p(artitions) q(uit)? d

type: MSCP
disk: RA92
label: SNOOPY
flags:
bytes/sector: 512
sectors/track: 73
tracks/cylinder: 13
```

```
sectors/cylinder: 949
cylinders: 3099
rpm: 3600
drivedata: 0 0 0 0 0

1 partitions:
#        size   offset    fstype   [fsize bsize]
  a: 2940951 0  2.11BSD    1024 1024    # (Cyl. 0 - 3098)

modify
```

Having now a disk with a label, we will need additional partitions for the core system, swapping etc. First of all we will change the partition `a` to be 100 MB in size:

```
d(isplay) g(eometry) m(isc) p(artitions) q(uit)? p
modify partitions
d(isplay) n(umber) s(elect) q(uit)? s
a b c d e f g h q(uit)? a
sizes and offsets may be given as sectors, cylinders
or cylinders plus sectors:  6200, 32c, 19c10s respectively
modify partition 'a'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? s
'a' size [2940951]: 204800
modify partition 'a'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? q
modify partitions
d(isplay) n(umber) s(elect) q(uit)? d

type: MSCP
disk: RA92
label: SNOOPY
flags:
bytes/sector: 512
sectors/track: 73
tracks/cylinder: 13
sectors/cylinder: 949
cylinders: 3099
rpm: 3600
drivedata: 0 0 0 0 0

1 partitions:
#        size   offset    fstype   [fsize bsize]
  a: 204800 0  2.11BSD    1024 1024     # (Cyl. 0 - 215*)

modify partitions
```

Now we need another partition, `b`, for swapping. This partition starts where partition `a` ends and will be 16 MB in size (a bit generous but having a disk as large as a `RA92` this should not be a problem):

```
d(isplay) n(umber) s(elect) q(uit)? s
a b c d e f g h q(uit)? b
```

```
sizes and offsets may be given as sectors, cylinders
or cylinders plus sectors:  6200, 32c, 19c10s respectively
modify partition 'b'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? o
'b' offset [0]: 204800
modify partition 'b'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? s
'b' size [0]: 32768
modify partition 'b'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? t
'b' fstype [unused]: swap
modify partition 'b'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? q
modify partitions
d(isplay) n(umber) s(elect) q(uit)? d

type: MSCP
disk: RA92
label: SNOOPY
flags:
bytes/sector: 512
sectors/track: 73
tracks/cylinder: 13
sectors/cylinder: 949
cylinders: 3099
rpm: 3600
drivedata: 0 0 0 0 0

2 partitions:
#        size   offset    fstype   [fsize bsize]
  a: 204800 0  2.11BSD    1024 1024     # (Cyl. 0 - 215*)
  b: 32768 204800  swap                 # (Cyl. 215*- 250*)

modify partitions
```

The next partition will be named `c` and covers the whole disk:

```
d(isplay) n(umber) s(elect) q(uit)? s
a b c d e f g h q(uit)? c
sizes and offsets may be given as sectors, cylinders
or cylinders plus sectors:  6200, 32c, 19c10s respectively
modify partition 'c'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? o
'c' offset [0]:
modify partition 'c'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? s
'c' size [0]: 2940951
modify partition 'c'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? t
'c' fstype [unused]:
modify partition 'c'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? q
modify partitions
d(isplay) n(umber) s(elect) q(uit)? d
```

```
type: MSCP
disk: RA92
label: SNOOPY
flags:
bytes/sector: 512
sectors/track: 73
tracks/cylinder: 13
sectors/cylinder: 949
cylinders: 3099
rpm: 3600
drivedata: 0 0 0 0 0

3 partitions:
#        size   offset    fstype   [fsize bsize]
  a: 204800 0  2.11BSD    1024 1024     # (Cyl. 0 - 215*)
  b: 32768 204800  swap                 # (Cyl. 215*- 250*)
  c: 2940951 0  unused    1024 1024     # (Cyl. 0 - 3098)

modify partitions
```

The next partition, g, will cover the remaining space of the drive – please note that the partition type has to be set to `2.11BSD`:

```
d(isplay) n(umber) s(elect) q(uit)? s
a b c d e f g h q(uit)? g
sizes and offsets may be given as sectors, cylinders
or cylinders plus sectors:  6200, 32c, 19c10s respectively
modify partition 'g'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? o
'g' offset [0]: 237568
modify partition 'g'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? s
'g' size [0]: 2703383
modify partition 'g'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? t
'g' fstype [unused]: 2.11BSD
modify partition 'g'
d(isplay) z(ero) t(ype) o(ffset) s(ize) f(rag) F(size) q(uit)? q
modify partitions
d(isplay) n(umber) s(elect) q(uit)? d

type: MSCP
disk: RA92
label: SNOOPY
flags:
bytes/sector: 512
sectors/track: 73
tracks/cylinder: 13
sectors/cylinder: 949
cylinders: 3099
rpm: 3600
drivedata: 0 0 0 0 0
```

```
7 partitions:
#        size   offset    fstype   [fsize bsize]
  a: 204800 0  2.11BSD    1024 1024     # (Cyl. 0 - 215*)
  b: 32768 204800  swap                 # (Cyl. 215*- 250*)
  c: 2940951 0  unused    1024 1024     # (Cyl. 0 - 3098)
  g: 2703383 237568  2.11BSD    1024 1024       # (Cyl. 250*- 3098*)

modify partitions
```

Now we are finished and can write the changes we did back to the disk:

```
d(isplay) g(eometry) m(isc) p(artitions) q(uit)? q
d(isplay) D(efault) m(odify) w(rite) q(uit)? w
d(isplay) D(efault) m(odify) w(rite) q(uit)? q

73Boot from ts(0,0,1) at 0172522
:
```

2.3 Creating and populating the root filesystem

After we have set up the system disk we can now create an empty root filesystem:

```
: ts(0,2)
Boot: bootdev=01002 bootcsr=0172522
Mkfs
file system: ra(0,0)
file sys size [102400]:
bytes per inode [4096]:
interleaving factor (m; 2 default):
interleaving modulus (n; 474 default):
isize = 25600
m/n = 2 474
Exit called

73Boot from ts(0,0,2) at 0172522
:
```

The next step is to copy the root filesystem image from tape to disk:

```
: ts(0,3)
Boot: bootdev=01003 bootcsr=0172522
Restor
Tape? ts(0,5)
Disk? ra(0,0)
Last chance before scribbling on disk. End of tape

73Boot from ts(0,0,3) at 0172522
:
```

2.4 Booting UNIX for the 1st time

Now we can boot the newly installed BSD image for the 1st time by loading the standalone boot program from `ra(0,0)`:

```
: ra(0,0)unix
Boot: bootdev=02400 bootcsr=0172150

2.11 BSD UNIX #115: Sat Apr 22 19:07:25 PDT 2000
    sms1@curly.2bsd.com:/usr/src/sys/GENERIC

ra0: Ver 3 mod 3
ra0: RA92  size=2940951

phys mem  = 3145728
avail mem = 2921792
user mem  = 307200

June  8 21:21:24 init: configure system

hk ? csr 177440 vector 210 skipped:  No CSR.
ht ? csr 172440 vector 224 skipped:  No CSR.
ra 0 csr 172150 vector 154 vectorset attached
rl 0 csr 174400 vector 160 attached
tm ? csr 172520 vector 224 does not exist.
tms 0 csr 174500 vector 260 vectorset attached
ts 0 csr 172520 vector 224 attached
xp 0 csr 176700 vector 254 attached
erase, kill ^U, intr ^C
#
```

Since the disk is currently not bootable we have to copy the proper bootstrap into block 0. All available bootstraps are located in `/mdec` – since we are using a `RA92` disk drive, we need `rauboot` which contains the `ra` driver (be sure to use the raw device of the destination disk):

```
# ls -la /mdec
total 16
drwxr-xr-x  2 root          512 Apr 23  2000 .
drwxr-xr-x 14 root          512 Apr 23  2000 ..
-r--r--r--  1 root          512 Dec  5  1995 bruboot
-r--r--r--  1 root          512 Dec  5  1995 dvhpuboot
-r--r--r--  1 root          512 Dec  5  1995 hkuboot
-r--r--r--  1 root          512 Dec  5  1995 hpuboot
-r--r--r--  1 root          512 Dec  5  1995 rauboot
-r--r--r--  1 root          512 Dec  5  1995 rkuboot
-r--r--r--  1 root          512 Dec  5  1995 rluboot
-r--r--r--  1 root          512 Dec  5  1995 rm03uboot
-r--r--r--  1 root          512 Dec  5  1995 rm05uboot
-r--r--r--  1 root          512 Dec  5  1995 rx01uboot
-r--r--r--  1 root          512 Dec  5  1995 rx02uboot
-r--r--r--  1 root          512 Dec  5  1995 si51uboot
-r--r--r--  1 root          512 Dec  5  1995 si94uboot
-r--r--r--  1 root          512 Dec  5  1995 si95uboot
# dd if=/mdec/rauboot of=/dev/rra0a count=1
1+0 records in
1+0 records out
#
```

We can now stop the simulator by pressing CTRL-E and boot from the newly created system disk:

```
Simulation stopped, PC: 004376 (MOV (SP)+,177776)
sim> b rq0

73Boot from ra(0,0,0) at 0172150
: ra(0,0,0)unix
Boot: bootdev=02400 bootcsr=0172150

2.11 BSD UNIX #115: Sat Apr 22 19:07:25 PDT 2000
    sms1@curly.2bsd.com:/usr/src/sys/GENERIC

ra0: Ver 3 mod 3
ra0: RA92  size=2940951

phys mem  = 3145728
avail mem = 2921792
user mem  = 307200

June  8 21:21:24 init: configure system

hk ? csr 177440 vector 210 skipped:  No CSR.
ht ? csr 172440 vector 224 skipped:  No CSR.
ra 0 csr 172150 vector 154 vectorset attached
rl 0 csr 174400 vector 160 attached
tm ? csr 172520 vector 224 does not exist.
tms 0 csr 174500 vector 260 vectorset attached
ts 0 csr 172520 vector 224 attached
xp 0 csr 176700 vector 254 attached
erase, kill ^U, intr ^C
#
```

2.5 Setting up /usr

Now we can proceed to setup the basic system:

```
# date 1112261655
date: can't write wtmp file.
Mon Dec 26 16:55:00 PST 2011
# passwd root
Changing password for root.
New password:
Retype new password:
# hostname snoopy
```

To setup the /usr filesystem we have to create a new filesystem in partition g and populate this filesystem with the contents of the save file on tape after mounting the partition on /usr:

```
# newfs ra0g
newfs: /sbin/mkfs -m 2 -n 474 -i 4096 -s 1351691 /dev/rra0g
isize = 65488
```

```
m/n = 2 474
# mount /dev/ra0g /usr
```

But first we have to create new device files for the `ts` tape drive:

```
# cd /dev
# rm *mt*
# ./MAKEDEV ts0
# sync
```

We can now, finally, start to copy the contents of `/usr` from tape to the newly mounted partition `g`:

```
# cd /usr
# mt rew
# mt fsf 6
# tar xpbf 20 /dev/rmt12
```

The next tape file contains the sources of the kernel and all necessary include files which will be restored to `/usr/src` (the `mt` command positions the tape to the beginning of the next tape file):

```
# mkdir src
# cd src
# mt -f /dev/rmt12 fsf
# tar xpbf 20 /dev/rmt12
```

We are nearing the end of the installation by now. The next step is to set the file protections on some central directories and create a symbolic link to the system sources:

```
# cd /
# chmod 755 / /usr /usr/src /usr/src/sys
# rm -f sys
# ln -s /usr/src/sys sys
```

Due to an acute paranoia attack we will check the file system integrity of partition `g` of the system disk:

```
# umount /dev/ra0g
# fsck /dev/rra0g
** /dev/rra0g
File System: /usr


NEED SCRATCH FILE (332 BLKS)
ENTER FILENAME:  /tmp/usrfsck
** Last Mounted on /usr
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
5139 files, 38744 used, 1308852 free
```

Everything seems to be correct, so we will remount `/usr` and add a line to `/etc/fstab` to make sure `/usr` will be mounted automatically during system startup:

```
# mount /dev/ra0g /usr
# cd /etc
# echo "/dev/ra0g       /usr    ufs     rw              1       1" >> fstab
# cat fstab
/dev/ra0a       /       ufs     rw              1       1
/dev/ra0b       none    swap    sw              0       0
/dev/ra0g       /usr    ufs     rw              1       1
#
```

We will now halt the BSD system and make a copy of the disk container file – just to make sure we have a good checkpoint in case something goes wrong:

```
# halt
syncing disks... done
halting

HALT instruction, PC: 000014 (MOV #1,17406)
sim> quit
alberich$ cp media/ra92.dsk ra92.dsk_bare_system
```

Now let us boot the system just assembled again (starting the emulator as shown in section 2.1):

```
sim> b rq0

73Boot from ra(0,0,0) at 0172150
: ra(0,0,0)unix
Boot: bootdev=02400 bootcsr=0172150

2.11 BSD UNIX #115: Sat Apr 22 19:07:25 PDT 2000
    sms1@curly.2bsd.com:/usr/src/sys/GENERIC

ra0: Ver 3 mod 3
ra0: RA92  size=2940951

phys mem  = 3145728
avail mem = 2921792
user mem  = 307200

December 26 17:19:10 init: configure system

hk ? csr 177440 vector 210 skipped:  No CSR.
ht ? csr 172440 vector 224 skipped:  No CSR.
ra 0 csr 172150 vector 154 vectorset attached
rl 0 csr 174400 vector 160 attached
tm ? csr 172520 vector 224 does not exist.
tms 0 csr 174500 vector 260 vectorset attached
ts 0 csr 172520 vector 224 attached
xp 0 csr 176700 vector 254 attached
erase, kill ^U, intr ^C
#
```

To leave the single user mode just press `CTRL-D` – the boot process will continue:

```
Fast boot ... skipping disk checks
checking quotas: done.
Assuming non-networking system ...
checking for core dump...
preserving editor files
clearing /tmp
standard daemons: update cron accounting.
starting lpd
starting local daemons: sendmail.
Mon Dec 26 17:19:16 PST 2011


2.11 BSD UNIX (curly.2bsd.com) (console)

login: root
Password:
erase, kill ^U, intr ^C
#
```

The tape image that comes with BSD 2.11 (instead of a real distribution on tapes which spans two tapes) contains the sources of all commands etc. in the last file on the tape. To extract these sources we will proceed as follows: First position the tape to the right file, then change the directory to `/usr/src` and extract the tape archive file's contents before finally stopping the system and ending the PDP11 emulation:

```
# mt fsf 8
# cd /usr/src
# tar xpb 20
# halt
syncing disks... done
halting

HALT instruction, PC: 000014 (MOV #1,17406)
sim> quit
Goodbye
alberich$
```

3 Configuring the system

The next big task to accomplish is the configuration of the system – this involves building a new kernel to include support for the devices of the target system, most notably the DELQA ethernet controller.

3.1 Compiling a new kernel

Let us start the emulator and boot the image just created again:

```
alberich$ ../../simh/pdp11

PDP-11 simulator V3.8-1
```

```
sim> set tto 7b
sim> cd cpu 3072k
Unknown command
sim> set cpu 3072k
Disabling CR
Disabling RK
Disabling HK
Disabling TM
sim> set rq0 ra92
sim> attach rq0 media/ra92.dsk
sim> boot rq0

73Boot from ra(0,0,0) at 0172150
: ra(0,0,0)unix
Boot: bootdev=02400 bootcsr=0172150

2.11 BSD UNIX #115: Sat Apr 22 19:07:25 PDT 2000
    sms1@curly.2bsd.com:/usr/src/sys/GENERIC

ra0: Ver 3 mod 3
ra0: RA92  size=2940951

phys mem  = 3145728
avail mem = 2921792
user mem  = 307200

December 26 17:29:37 init: configure system

hk ? csr 177440 vector 210 skipped:  No CSR.
ht ? csr 172440 vector 224 skipped:  No CSR.
ra 0 csr 172150 vector 154 vectorset attached
rl 0 csr 174400 vector 160 attached
tm ? csr 172520 vector 224 skipped:  No CSR.
tms 0 csr 174500 vector 260 vectorset attached
ts ? csr 172520 vector 224 skipped:  No CSR.
xp 0 csr 176700 vector 254 attached
erase, kill ^U, intr ^C
# cd /usr/src/sys/conf
/usr/src/sys/conf: bad directory
# Fast boot ... skipping disk checks
checking quotas: done.
Assuming non-networking system ...
checking for core dump...
preserving editor files
clearing /tmp
standard daemons: update cron accounting.
starting lpd
starting local daemons: sendmail.
Mon Dec 26 17:31:42 PST 2011


2.11 BSD UNIX (curly.2bsd.com) (console)

login: root
```

Board	Device	CSR address	Vector
M8061	RL controller	774400	160
Emulex	TS tape controller	772520	224
M7516	DELQA	774440	120
3rd party	RX02 controller	777170	264
M7957	DZV11 terminal multiplexer	760100	300
Emulex	Dual SCSI controller	772150	154
		760354	310

Table 1: Device configuration of the PDP11

```
Password:
erase, kill ^U, intr ^C
#
```

The machine has also an EMULEX SCSI controller card, a `DZV11` terminal multiplexer[1] and a `DELQA` ethernet controller etc. The overall configuration is shown in table 1. We will now create a new configuration file for the new kernel to be built.

These configuration files are found in `/usr/src/sys/conf` – we will make a copy of `GENERIC`, name it `SNOOPY` and perform the necessary changes to that file:

```
# cd /usr/src/sys/conf
# cp GENERIC SNOOPY
```

These things have to be changed:

- Set `LINEHZ` to `50` instead of `60`, reflecting the European line frequency.
- Set `PDP` to `73` instead of `GENERIC` since this represents our system.
- Set `IDENT` to `SNOOPY` and `MAXUSERS` to `8`.
- The filesystem configuration is a bit more tricky – since we use a MSCP device (a SCSI disk on an Emulex controller) we have to change the variables `PIPEDEV`, `ROOTDEV` and `SWAPDEV` accordingly. Therefore the offset of the `ra` driver in the `bdevsw` table in the file `/usr/src/sys/pdp/conf.c` is necessary. This number is `5`, so we will set `PIPEDEV` and `ROOTDEV` to `makedev(5,0)` and `SWAPDEV` to `makedev(5,1)`.
- The only change in the kernel configuration is setting `SOFUB_MAP` to `0` since we do not need a software UNIBUS/QBUS map that would allow us to use 18 bit controllers in a 22 bit QBUS system[2].
- Next we will configure the available disk drives: Set `NBR` to `0`, `NHK` to `0`, `NRAC` to `2`, `NRAD` to `4`, `NRK` to `0`, `NRL` to `2`, `NRX` to `2` and `NSI`, `NXPC`, `NXPD`, `NRAM` to `0`.
- Configuring the tape drives: Set `NHT`, `NTM` to `0`, set `AVIVTM` to `NO`, set `NTS` to `1` (this is our `TS11` tape drive), set `NTMSCP` and `NTMS` to `0`, leave `TMSCP_DEBUG` set to `0`.

[1] According to [LSI-11 SSM][Appendix B, p. 1125], the `DZV11` is the QBUS analog of the UNIBUS `DZ11`.

[2] This parameter is set to `YES` for the `GENERIC` kernel.

- Now for the terminals: Leave `NKL`, the number of `DL11s`, set to `1`[3], set `NDZ` to `1`[4].
- To configure network support, set `INET` to `YES` and `NETHER` to `1`. The `GENERIC` kernel does not need any pseudo terminals but since some applications require them we will create 10 pseudo terminals[5] by setting `NPTY` to `10`. To support our `DELQA` ethernet controller, we set `NQE` to `1i`[6].

Having created a new configuration file named `SNOOPY` like this we can no compile a new kernel for the system:

```
# ./config SNOOPY
Creating ../SNOOPY.
Copying standard files to ../SNOOPY.
Setting configuration options for SNOOPY.
Creating device header files.
Creating Makefile for SNOOPY.
# cd /sys/SNOOPY
# make
```

The `make` command produces lots of output (and it is really a good idea to run all of this on the emulator instead of a real PDP11 since this operation takes ages on a real PDP11 :-)). The only problem is that this kernel build will fail since the linker will be unable to pack everything we requested into the address space available on a PDP11 (64 kB minus 8 kB used for IO space):

```
ld -q -r -d -X -i -o unix.o scb.o  mch_backup.o mch_click.o ...
ld: too big for type 431
*** Exit 2

Stop.
#
```

The type `431` denotes a split I/D space with overlays – a BSD 2.11 kernel for a PDP11 requires split I/D space and overlays, so this is OK[7]. Let us have a detailed look at `unix.o`:

```
# size unix.o
text    data    bss     dec     hex
58880   6828    24450   90158   1602e   total text: 110208
        overlays: 7744,7360,7872,7296,2240,8384,4864,5568
#
```

Now it is getting a bit messy: The address space of a PDP11 is 64 kB of which 8 kB are reserved for IO devices. How does BSD 2.11 handle a kernel larger than the remaining 56 kB (including all necessary data structures)? It uses overlays of 8 kB each, so the idea is to have a base segment being 48 kB in size with various overlays containing the modules that did not fit into the base segment. Getting the

[3] The console terminal is a `DL11`.

[4] Although our the `DZV11` only supports four instead of eight serial lines.

[5] Note that each `PTY` requires 78 bytes in the kernel data space!

[6] Althoughi `NQT` might seem more appropriate, it is for a late `DELQA` only.

[7] There is a great news article describing this problem and its solution in detail: `http://www.dnull.com/bsd/oldnews/bsdnew62161.html`.

base segment below 48 kB is simple – just move modules to an overlay segment (as long as they fit). This is done in the `Makefile`. The section starting with `BASE=` contains the list of all modules that go into the base segment. Following this section are up to 15 overlay segments `OV1` to `OV15`. Please note that no overlay segment may be empty except the last ones! After some severe fiddling with `Makefile` the relevant portions look like this:

```
BASE=   br.o clock.o cons.o dh.o dhu.o dhv.o dr.o dz.o hk.o ht.o init_sysent.o \
        kern_clock.o kern_descrip.o kern_mman.o kern_proc.o kern_prot.o \
        kern_prot2.o kern_subr.o kern_synch.o lp.o machdep.o ra.o ram.o \
        rk.o rl.o rx.o si.o subr_rmap.o sys_inode.o sys_kern.o \
        tm.o ts.o tty.o tty_conf.o tty_subr.o tty_tb.o \
        ufs_bmap.o ufs_inode.o kern_xxx.o subr_xxx.o \
        vm_proc.o vm_sched.o vm_swap.o xp.o quota_subr.o
OV1=    sys_generic.o ufs_syscalls.o vfs_vnops.o
OV2=    kern_acct.o kern_exec.o kern_exit.o kern_fork.o kern_resource.o
OV3=    kern_time.o sys_process.o ufs_mount.o ufs_subr.o uipc_syscalls.o
OV4=    dkbad.o kern_sig.o mem.o trap.o tty_pty.o tty_tty.o
OV5=    quota_kern.o quota_ufs.o quota_sys.o \
        sys_pipe.o
OV6=    ufs_disksubr.o ufs_dsort.o ufs_syscalls2.o kern_sig2.o
OV7=    mch_fpsim.o kern_sysctl.o ingreslock.o vm_text.o
OV8=    ufs_bio.o ufs_namei.o
# OV9 gets the (hopefully) never used routines
OV9=    dn.o init_main.o kern_pdp.o machdep2.o subr_prf.o syscalls.o \
        subr_log.o vm_swp.o
OV10=   tmscp.o tmscpdump.o toy.o ufs_alloc.o ufs_fio.o
```

Running `make` again works fine now:

```
# make
... ... ... ...
size unix
text    data    bss     dec     hex
50048   6830    24450   81328   13db0   total text: 110528
        overlays: 7744,7360,7872,7232,1344,4864,5568,8000,7872,2624
Compacting symbol table
symcompact unix
symcompact: 252 symbols removed
Compacting strings table
strcompact unix
rearranging symbols
symorder ../pdp/symbols.sort unix
./checksys unix
System will occupy 200992 bytes of memory (including buffers and clists).

               end {0075060}          nbuf {0015214}           buf {0047244}
             nproc {0015202}          proc {0060760}         ntext {0015204}
              text {0074020}         nfile {0015210}          file {0071364}
            ninode {0015206}         inode {0015400}      ncallout {0015212}
           callout {0037010}     ucb_clist {0015220}        nclist {0015216}
          ram_size {0000000}       xitdesc {0015376}      quotdesc {0000000}
         namecache {0046742}       _iosize {0014036}          nlog {0014510}
ld -X -i -o netnix net_copy.o net_csv.o  net_mbuf.o net_scb.o ...
```

```
size netnix
text    data    bss     dec     hex
59776   2314    38906   100996  18a84
Compacting symbol table
symcompact netnix
symcompact: 247 symbols removed
Compacting strings table - this will take a few minutes
strcompact netnix
rearranging symbols
symorder ../pdp/symbols.sort netnix
#
```

Now let us save the old kernel (just in case we messed up), install the new one and reboot the system to test it:

```
# cp /unix /oldunix
# make install
install -c -o root -g kmem -m 744 unix /unix
install -c -o root -g kmem -m 744 netnix /netnix
# halt
syncing disks... done
halting

HALT instruction, PC: 000014 (MOV #1,17406)
sim> b rq0

73Boot from ra(0,0,0) at 0172150
: ra(0,0,0)unix
Boot: bootdev=02400 bootcsr=0172150

2.11 BSD UNIX #26: Mon Dec 26 19:10:23 PST 2011
    root@curly.2bsd.com:/usr/src/sys/SNOOPY

ra0: Ver 3 mod 3
ra0: RA92  size=2940951
attaching qe0 csr 174440
qt@174440 !-YM
attaching lo0

phys mem  = 3145728
avail mem = 2808384
user mem  = 307200

December 26 19:13:36 init: configure system

hk ? csr 177440 vector 210 skipped:  No autoconfig routines.
ht ? csr 172440 vector 224 skipped:  No autoconfig routines.
ra 0 csr 172150 vector 154 vectorset attached
rl 0 csr 174400 vector 160 attached
tm ? csr 172520 vector 224 skipped:  No autoconfig routines.
tms ? csr 174500 vector 260 skipped:  No autoconfig routines.
ts ? csr 172520 vector 224 skipped:  No CSR.
xp ? csr 176700 vector 254 skipped:  No autoconfig routines.
erase, kill ^U, intr ^C
```

```
# Fast boot ... skipping disk checks
checking quotas: done.
Assuming NETWORKING system ...
ifconfig: ioctl (SIOCGIFFLAGS): no such interface
add host curly.2bsd.com: gateway localhost.2bsd.com
add net default: gateway 206.139.202.1: Network is unreachable
starting system logger
Dec 26 19:14:01 curly vmunix: ra0: Ver 3 mod 3
Dec 26 19:14:01 curly vmunix: ra0: RA92  size=2940951
checking for core dump...
preserving editor files
clearing /tmp
standard daemons: update cron accounting.
starting network daemons: inetd rwhod printer.
starting local daemons: sendmail.
Mon Dec 26 19:14:01 PST 2011
Dec 26 19:14:01 curly December 26 19:14:01 init: kernel security level changed from 0 to 1


2.11 BSD UNIX (curly.2bsd.com) (console)

login: root
Password:
erase, kill ^U, intr ^C
#
```

3.2 Devices...

The next step involves the creation of devices for our current kernel:

```
# cd /dev
# ./MAKEDEV std local ra0 ra1 ra2 ra2 mt0 nmt0 rl0 rl1 pty0 rz0
```

Now make sure that the CSR- and VEC-addresses of the devices configured in the system are correct. These settings are done in `/etc/dtab`. The only things which have to be changed here are the lines for the `dz` terminal multiplexer and the `rx` floppy controller which have to be activated by removing the comment character in column 1. In addition to that the entry for the ethernet controller (`qe`) has to be activated and its VEC address has to be changed from `400` to `120`. To avoid unnecessary warnings during system startup, the entries for `hk`, `kt`, `tm` and `tms` have been disabled by commenting them out.

In addition to that `/etc/MAKEDEV.local` has to be adapted to the current environment. Most notably we have to make sure that the swap device is linked to the correct partition, in this case to `/dev/ra0b`:

```
rm -f swap;             ln ra0b swap
chgrp kmem swap
chmod 640 swap
```

Also, we should perform a `cd /etc; ./MAKEDEV kmem`.

3.3 Configuring the network

Edit `/etc/hosts` to suit the local environment:

```
127.0.0.1       localhost.2bsd.com localhost.arpa localhost
192.168.31.16   fafner.pi-research.de fafner
192.168.31.62   snoopy.pi-research.de snoopy
```

...and adapt `/etc/netstart` accordingly by changing the variable `hostname` to contain the value `snoopy.pi-research.de`. In addition to that `/etc/printcap` has to be edited, too (I just removed any entries by commenting them out). One should also not forget to adapt `/etc/ntp.conf`.

4 Getting it to the real PDP11

Now we have to copy the disk image to a real disk and attach this to the destination system, the Frankenstein PDP11 which is shown in figure 1. This machine is really ugly but fast – and it is a bit strange: Obviously it is a QBUS machine in a BA23 enclosure but the CPU card is from a PDP11/84 (unfortunately that was the only thing I could save from scrap from this 11/84 – I found it on the office wall of someone who found this board to pretty to be scrapped – this previous owner one day gave it to me), so the machine thinks it is a PDP11/84 (with FPU chip, by the way!) while it is in fact a QBUS machine. The top drawer contains a RX02 compatible diskette drive, followed by a Cipher front loading tape drive, the CPU enclosure and two RL02 disk drives (10MB each)[8].

4.1 Writing the image to a real disk

The only other system that is currently running having a SCSI controller is FAFNER, my VAX 7000/820, so copying the disk image contaning the BSD 2.11 UNIX involves the following steps:

1. Transfer the image to the VAX using `ftp`[9].
2. Copy the image to a foreign mounted disk.

```
alberich$ ftp fafner
Connected to fafner.
220 fafner.pi-research.de FTP Server (Version 5.3) Ready.
Name (fafner:ulmann):
331 Username ulmann requires a Password
Password:
cd bast230 User logged in.
Remote system type is VMS.
ftp> cd bastel
250-CWD command successful.
250 New default directory is DISK$USER_0:[ULMANN.BASTEL]
ftp> bin
200 TYPE set to IMAGE.
ftp> put ra92.dsk
```

[8] As you will have realized the rack is as much non-DEC as possible – it is from an old Honeywell installation and was the only empty rack I had at home when I started to put this PDP11 together from the various parts described.

[9] Using a NFS share would be more elegant but the NFS server of TCPIP 5.3 is too buggy to attempt a transfer as large as this.

Figure 1: The destination system – a QBUS PDP11 that thinks it is a PDP11/84

```
local: ra92.dsk remote: ra92.dsk
229 Entering Extended Passive Mode (|||61798|)
150 Opening data connection for DISK$USER_0:[ULMANN.BASTEL]RA92.DSK; (192.168.31.18,60363)
226 Transfer complete.
1505682432 bytes sent in 46:31 (526.75 KB/s)
ftp> bye
221 Goodbye.
alberich$ telnet fafner
Trying 192.168.31.16...
Connected to fafner.
Escape character is '^]'.
Username: system
Password:
        Welcome to OpenVMS VAX V7.2

    Last interactive login on Monday, 26-DEC-2011 17:46
    Last non-interactive login on Monday, 26-DEC-2011 17:46

SYSTEM:FAFNER$ mou/for $1$dka4
%MOUNT-I-MOUNTED, SCRATCH mounted on _$1$DKA4: (HSJ0)
SYSTEM:FAFNER$ copy/log disk$user_0:[ulmann.bastel]ra92.dsk $1$dka4:
%COPY-S-COPIED, DISK$USER_0:[ULMANN.BASTEL]RA92.DSK;1 copied to $1$DKA4:[]RA92.D
SK;1 (2940786 blocks)
SYSTEM:FAFNER$ dism $1$dka4
SYSTEM:FAFNER$ lo
Connection closed by foreign host.C-2011 19:27:11.67
alberich$
```

This disk is then (physically) transfered to the PDP11. Here is the log of the first boot on the real machine:

```
Testing in progress - Please wait
Memory Size is 4088 K Bytes
9 Step memory test
  Step 1 2 3 4 5 6 7 8 9


Message 04      Entering Dialog mode

Commands are Help, Boot, List, Setup, Map and Test.
Type a command then press the RETURN key: B DU0


Trying DU0

Starting system from DU0


83Boot from ra(0,0,0) at 0172150
: ra(0,0,0)unix
Boot: bootdev=02400 bootcsr=0172150

2.11 BSD UNIX #26: Mon Dec 26 19:10:23 PST 2011
    root@curly.2bsd.com:/usr/src/sys/SNOOPY
```

```
ra0: Ver 5 mod 13
ra0: RA81  size=4110393
attaching qe0 csr 174440
qt@174440 !Turbo
attaching lo0

phys mem  = 4186112
avail mem = 3848768
user mem  = 307200

December 26 20:57:51 init: configure system

dz 0 csr 160100 vector 310 attached
ra 0 csr 172150 vector 154 vectorset attached
rl 0 csr 174400 vector 160 attached
rx 0 csr 177170 vector 264 attached
ts 0 csr 172520 vector 224 attached
qe ? csr 174440 vector 120 skipped:  No autoconfig routines.
erase, kill ^U, intr ^C
# Fast boot ... skipping disk checks
checking quotas: done.
/dev/swap: Operation not supported by device
Assuming NETWORKING system ...
add host snoopy.pi-research.de: gateway localhost.2bsd.com
add net default: gateway 192.168.31.254
starting system logger
checking for core dump...
Dec 26 20:59:06 snoopy vmunix: ra0: Ver 5 mod 13
Dec 26 20:59:06 snoopy vmunix: ra0: RA81  size=4110393
preserving editor files
clearing /tmp
standard daemons: update cron accounting.
starting network daemons: inetd rwhod printer.
starting local daemons: sendmail.
Mon Dec 26 20:59:24 PST 2011
Dec 26 20:59:25 snoopy December 26 20:59:25 init: kernel security level changed from 0 to 1


2.11 BSD UNIX (snoopy.pi-research.de) (console)

login: root
Password:
erase, kill ^U, intr ^C
#
```

We now have a BSD 2.11 UNIX up and running on a real PDP11[10].

[10] Oh dear – I had almost forgotten how slow a real PDP11 is, compared with simh on a modern multi GHz processor, but there is nothing to compete with a *real* PDP11! :-)

5 Final configuration tasks

Now that the BSD 2.11 is happily running on a real PDP11 there are a few things left to do[11] which will be described in the following.

5.1 Serial lines

Since this particular PDP11 has a DZV11 quad terminal interface and since I do not want to use modem control[12] the terminal devices have to be created by hand:

```
# cd /etc
# mknod tty00 c 2 128
# mknod tty01 c 2 129
# mknod tty02 c 2 130
# mknod tty03 c 2 131
```

The trick are the minor device numbers which normally would have been `0` to `3` for these four asynchronous lines. Setting bit `7` turns on *soft carrier*, so these devices won't wait for a Carrier-Detect signal to respond to. To enable these serial lines for logins, `/etc/ttys` must be edited to contain the following lines:

```
tty00   "/usr/libexec/getty std.9600"   vt100           on secure
tty01   "/usr/libexec/getty std.9600"   vt100           on secure
tty02   "/usr/libexec/getty std.9600"   vt100           on secure
tty03   "/usr/libexec/getty std.9600"   vt100           on secure
```

5.2 User accounts

Setting up users on such a historic UNIX is quite straight forward. All that has to be done is to add an entry to `/etc/master.passwd`, copy this file to `/etc/passwd`, transform this into a valid `passwd` file and create a home directory. All users will belong to the group `staff` which has the GID `10`, and UIDs will start at `100`. So first add the following line to `/etc/master.passwd`:

```
ulmann::100:10::::Bernd Ulmann:/usr/home/ulmann:/bin/sh
```

No perform the copy, create a home directory etc. – the last step has to be done only once, of course:

```
# cp master.passwd passwd
# mkpasswd passwd
# mkdir -p /usr/home/ulmann
# chgrp staff /usr/home/ulmann
# chown ulmann /usr/home/ulmann
# chmod a+r /usr/home
```

This newly created user can now login without password – a password should be set immediately using the `passwd` command. If the user should be able to perform a `su`, the file `/etc/group` has to be adapted like this:

```
wheel:*:0:root,ulmann
```

[11] Thanks for `http://www.retrocmp.com/how-tos/installing-211bsd-unix-on-pdp-1144/` for a lot of valuable hints! :-)

[12] Modem control can be a real problem – without a proper Carrier-Detected signal no connection will be established and I want to use the four serial lines with simple three wire connections...

References

[2.11BSD] Stevan Schultz, *Installing and Operating 2.11BSD on the PDP-11*, June 13, 1995

[LSI-11 SSM] DEC, *LSI-11 Systems Service Manual*, three volumes, 5th edition, 1985