NetBSD and handheld platforms

Valeriy Ushakov <uwe@NetBSD.org>
Alistair Crooks <agc@NetBSD.org>
The NetBSD Project

Abstract

NetBSD has long been known for its portability to other platforms. The lower end of this range of platforms has included thin clients, and, most recently, handheld PCs. These machines are typically powered by a low-power CPU, and have smaller LCD screens than a laptop, and consequently dissipate less power. These machines usually use a touch screen and a stylus for a pointing device, can run for nearly a day on battery power only, and come with some version of Windows in ROM for an operating system.

This paper discusses a number of issues related to porting NetBSD to handheld PCs – the challenges, and the parts which are needed, including the subsystems still to be written. It contrasts the various handheld PC devices on the market, and looks at Linux support for these devices, as well as Windows and NetBSD. It also looks at the challenges of handheld and pocket PCs, from fitting a graphical browser onto a limited size LCD screen, to window managers designed for mouse-equipped environments, to user interface issues when no keyboard is present.

Finally, it looks into the future, and discusses what the ultimate geek gadget PDA would be.

Introduction

The NetBSD operating system has been ported to a large number of processor families and architectures: computers built around those CPUs range from high-end servers to small computers like laptops, notebooks, and, now, PDAs, handheld PCs (H/PC) and pocket PCs (PPC).

There are two aspects to porting NetBSD to a new platform:

- Porting to the new platform itself. This is occasionally referred to as the “Because it's there” question. There are technical challenges to bringing up an operating system on a new platform, and these are often the things that spur developers to take this lonely road.
- Once NetBSD has been ported to a new platform, the next challenge is to run NetBSD on that platform, from porting third-party applications which may only have been written with Linux and IA32/i386 architecture in mind, to the pure delight of debugging some of those same applications. However, this is also the stage in which the proselytizing of the platform, of NetBSD, and of the combination takes place.

At various conferences and trade shows, the handheld platforms have been the ultimate “geek gadget”, which has garnered a large amount of interest, especially for its size. Of course, the size, or lack of it, may be the reason for the interest – being able to use on-line services such as IRC and electronic mail from a handheld computer with a wireless card is a tremendous selling point, both for the platform and for NetBSD.

Target Platforms

All of the supported platforms are originally designed to run some version of Microsoft Windows CE. Most future platforms are likely be Windows CE based as well.[1]

Terminology

Microsoft terminology in this area and their different version numbering schemes are very confusing. The primary distinction that matters for our paper is:

- H/PC – Handheld PC. Keyboard, half or full size VGA display.
- PPC – Pocket PC. No built-in keyboard, quarter VGA display.

In general, we shall use the term "Personal Digital Assistants" (PDAs) throughout this paper to cover both HPC and PPC.

Lack of built-in keyboard might prove to be a big problem for standalone interactive use, unless something like QTopia or GPE is ported. However even PPC can be useful, perhaps in combination with a third-party package like xscribble (pkgsrc/x11/xscribble) which allows a user of a touch screen to input characters into X11 applications, using a uni-stroke (graffiti-like) alphabet – it uses the Xtest extension to allow synthesis of characters as though they had been typed on a keyboard.

Uses

HPCs are very useful as a mobile Unix terminal – they provide all the nice features of “big” NetBSD: IPv6, IPSEC, WiFi, SSH, etc… E.g., one of the authors used his Jornada for aiming a polarized WiFi aerial on a building’s rooftop. Laptop is just too bulky for that. PPCs, unfortunately, are not usable for this currently (need QTopia/GPE).

PDAs can be used as a development platform – if you develop software for an embedded platform with limited capabilities you can use NetBSD host as a development, debugging, and testing machine. This use is probably most important for SuperH-based PDAs, as “bigger” ARM and MIPS boxes are more easily available. Even for ARM and MIPS it might be cheaper to buy a handheld then a big machine. Even keyboardless PPC are suitable for this purpose.

These handhelds can be used as a PDA, bereft of Microsoft Office software (although Windows CE Office components tend to be simplified and feature-lacking versions of the traditional Microsoft Office components). Calendar, diary/agenda, spreadsheet, presentation graphics, etc can all be done by using open source equivalents. Insert a broad hint to pkgsrc folks here :)

As a musical juke box, an alternative to an iPod or a Creative Nomad, for playing MP3 or ogg files, again using traditional open source software to achieve this (it should be noted that some of the versions of Windows Media Player on PDAs are burned into ROM, and so upgrades are not usually an option – and early WMP have problems with VBR on some MP3 tracks).

By using the onboard Infra-red port, or USB 1.1 client, or onboard modem, the PDA can be used as a GPS receiver, as well as having all of the benefits above.

Booting NetBSD

There are various ways which different operating systems can boot on these devices. Usually, they will involve installing the alternative operating system onto a secondary storage

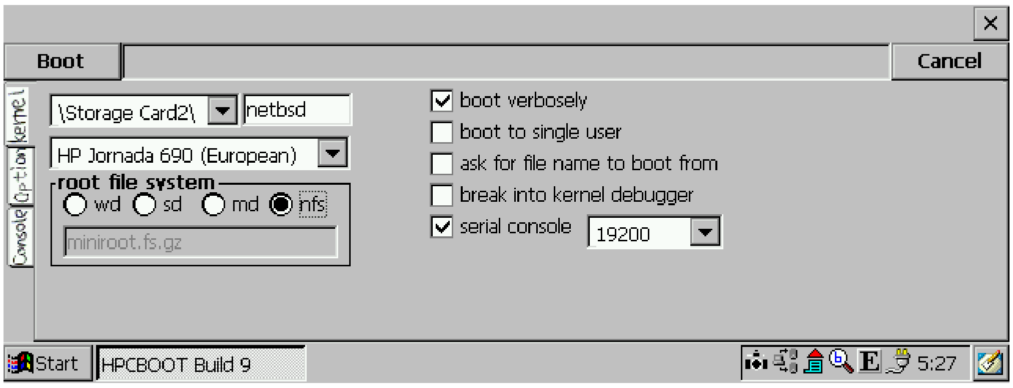
[1] There’s also Symbian OS, but devices based on it are not as widespread. Even the recent sub-notebook from Psion runs Windows CE.NET. We will not discuss Symbian based devices in this paper.

medium, such as a Compact Flash (CF) card - these are easy to find, have a large capacity, and are inexpensive; prices have fallen over the last two years.

Typically, HPC and PPC machines have ROM, where the "native" operating system will reside – this is typically Windows CE. Windows CE executes directly from ROM, thereby leaving all available RAM free for applications and data. It is only really "booted" after a hard reset. When suspended, it simply powers down all the devices and only spends a tiny bit of power on refreshing the RAM.

Some of these machines, such as the iPAQ, have flashable ROMs, whilst others have non-rewritable ROMs (such as the Jornada 600 and 700 series). Linux takes advantage of rewritable ROMs, and so it's possible to flash Linux into the ROM on an iPAQ. At the present time, NetBSD does not take advantage of this.

The NetBSD boot loader is, therefore, a separate program, and typically resides on Compact Flash card or some other method of persistent storage, accessed by the native operating system. This boot loader is then executed under the native operating system. The following screenshots show hpcboot, a Windows CE program, being used to boot NetBSD on a Jornada 690.



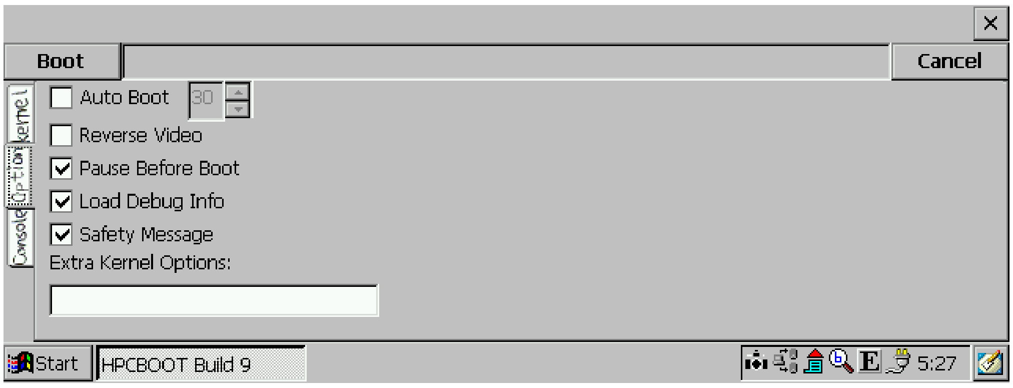
\Storage Card2\ netbsd
HP Jornada 690 (European)
root file system: ( ) wd ( ) sd ( ) md (●) nfs
miniroot.fs.gz

- [x] boot verbosely
- [ ] boot to single user
- [ ] ask for file name to boot from
- [ ] break into kernel debugger
- [x] serial console 19200

Figure 1. Selecting the kernel to boot and boot options

The figure above shows the "Kernel" tab of the hpcboot GUI, where the machine model is selected, the kernel to boot and the type of the root file system are specified, and boot options that can be entered.

The next figure shows the "Option" dialog, that controls hpcboot operation. The options selected on this screenshot are good defaults.



- [ ] Auto Boot 30
- [ ] Reverse Video
- [x] Pause Before Boot
- [x] Load Debug Info
- [x] Safety Message

Extra Kernel Options:

Figure 2. Options that control hpcboot operation

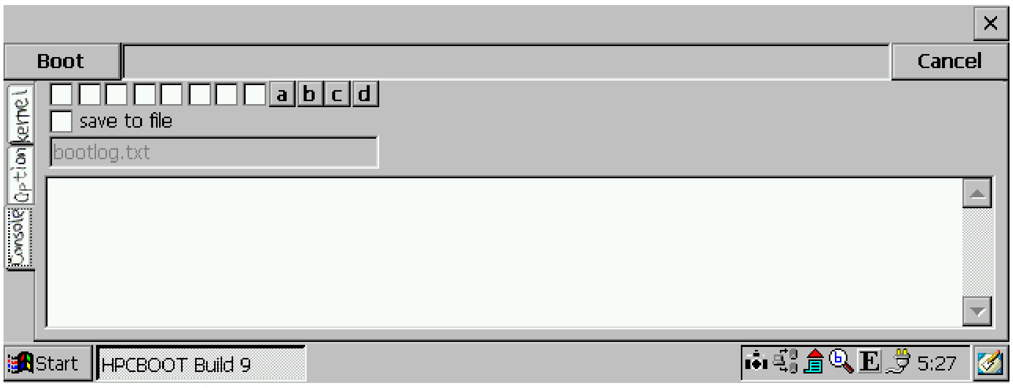


Figure 3. Output from hpcboot and debugging options

Finally, the last tab, “Console”, features a large text area where hpcboot reports gory details of its progress. There are also some anonymous buttons and checkboxes that are intended to be used for hpcboot debugging.

When NetBSD boots the Windows CE in your ROM is safe, however all your data, installed programs, etc will be lost. When NetBSD is shut down, Windows CE boots back as if after hard reset.

An arrangement that one of the authors finds convenient is as follows:

- Partition the CF card into an MSDOS partition, and a NetBSD partition (you need to do this on a Unix host). Your NetBSD installation will reside in the NetBSD partition of the CF. More on this later.
- After a hard reset (you have already tried to boot NetBSD and lost all your data on the device already, haven't you? ;), install all the Windows CE programs and drivers that you need, configure the system to suit your needs (e.g. dialup settings), then do a full backup. Put that full backup onto the DOS partition of your CF card. Saving another backup copy somewhere else is a good idea as well. Add your PIM (personal information manager) data, and then do a PIM-only backup. Put the PIM backup onto the DOS partition of your CF card as well.

  Make sure that you use the Windows CE backup program that is in your ROM! When NetBSD is shut down, the Windows CE comes up from hard reset and no fancy add-on backup programs that you have installed are available.
- Put the hpcboot.exe onto the DOS partition of the CF – you will need it to boot the NetBSD.
- And the last piece you need on the DOS partition is the NetBSD kernel. Hpcboot can boot kernels from the UFS partition, but having the kernel on the DOS partition is convenient when you want to update it, or try out a new kernel – you can write new kernel to the CF from any operating system, including the Windows CE itself.

So when planning the CF partitioning, you should take into account the space requirements for the things listed above. It is prudent to make a backup in advance to see how much space is used. It is advisable to have space for at least a couple of kernels and a miniroot image, and to reserve some space for files that you want to move between Windows and NetBSD. A partition of 16MB should be sufficient.

Installing NetBSD

So how would you install the NetBSD to the CF card? The standard NetBSD system installer, sysinst, is not yet supported on HPC platforms. Adding HPC support to sysinst is really a SMOP, but nobody has done the legwork yet. Thus, currently the easiest way is to use another NetBSD host to do the installation. Hint: your HPC booted with root file system on NFS qualifies!

If you have an i386 laptop running NetBSD you can connect your CF card to it using either a USB flash card reader, or a CF to PCMCIA adaptor.

Before we proceed with a detailed walkthrough, here is the outline of the process:

- Partition the CF card into DOS and NetBSD partitions (see previous section).
- Disklabel the NetBSD partition.
- Create DOS and UFS file systems.
- Mount both partitions.
- Transfer hpcboot and kernel to the DOS file system.
- Extract installation sets to the NetBSD file system.
- Edit several files in `/etc` to pre-configure your system.
- Move the CF to the HPC and boot!

Installation Walkthrough

This installation walkthrough is based on the transcript taken during a from-scratch installation of NetBSD/hpcsh.

Here is the CF card that we will do our sample installation onto:

```
wdc0 at pcmcia0 function 0
atabus2 at wdc0 channel 0
wd1 at atabus2 drive 0: <Transcend 256M>
wd1: drive supports 1-sector PIO transfers, LBA addressing
wd1: 244 MB, 978 cyl, 16 head, 32 sec, 512 bytes/sect x 500736 sectors
wd1: drive supports PIO mode 4
```

We start by splitting the CF card into a small DOS partition and a NetBSD partition. It’s important that the DOS partition comes first. Windows get very confused otherwise.

```
# fdisk -u wd1
Disk: /dev/rwd1d
NetBSD disklabel disk geometry:
cylinders: 978, heads: 16, sectors/track: 32 (512 sectors/cylinder)
total sectors: 500736

BIOS disk geometry:
cylinders: 978, heads: 16, sectors/track: 32 (512 sectors/cylinder)
total sectors: 500736

Do you want to change our idea of what BIOS thinks? [n] <enter>

Partition table:
0: Primary 'big' DOS, 16-bit FAT (> 32MB) (sysid 6)
    start 32, size 500192 (244 MB, Cyls 0-977), Active
1: <UNUSED>
2: <UNUSED>
3: <UNUSED>
Which partition do you want to change?: [none] 0
The data for partition 0 is:
Primary 'big' DOS, 16-bit FAT (> 32MB) (sysid 6)
    start 32, size 500192 (244 MB, Cyls 0-977), Active
sysid: [0..255 default: 6] 1
start: [0..978cyl default: 32, 0cyl, 0MB] <enter>
size: [0..978cyl default: 500192, 977cyl, 244MB] 14MB
```

```
bootmenu: [] **<enter>**
The bootselect code is not installed, do you want to install it now? [n] **<enter>**

Partition table:
0: Primary DOS with 12 bit FAT (sysid 1)
    start 32, size 28640 (14 MB, Cyls 0-56), Active
1: <UNUSED>
2: <UNUSED>
3: <UNUSED>
Which partition do you want to change?: [none] **1**
The data for partition 1 is:
<UNUSED>
sysid: [0..255 default: 169]  **<enter>**
start: [0..978cyl default: 28672, 56cyl, 14MB] **<enter>**
size: [0..922cyl default: 472064, 922cyl, 231MB] **<enter>**
bootmenu: [] <enter>
The bootselect code is not installed, do you want to install it now? [n] **<enter>**

Partition table:
0: Primary DOS with 12 bit FAT (sysid 1)
    start 32, size 28640 (14 MB, Cyls 0-56), Active
1: NetBSD (sysid 169)
    start 28672, size 472064 (231 MB, Cyls 56-978)
2: <UNUSED>
3: <UNUSED>
Which partition do you want to change?: [none] **<enter>**

We haven't written the MBR back to disk yet.  This is your last chance.
Partition table:
0: Primary DOS with 12 bit FAT (sysid 1)
    start 32, size 28640 (14 MB, Cyls 0-56), Active
1: NetBSD (sysid 169)
    start 28672, size 472064 (231 MB, Cyls 56-978)
2: <UNUSED>
3: <UNUSED>
Should we write new partition table? [n] **y**
```

Now that partitions are ready we need to edit the NetBSD disklabel. We start by using mbrlabel(8) that can update a NetBSD disk label from the Master Boot Record (MBR) label.

```
# **mbrlabel wd1**
Found MSDOS partition; size 28640 (13 MB), offset 32
  skipping existing MSDOS partition at slot e.
Found 4.2BSD partition; size 472064 (230 MB), offset 28672
  skipping existing unused partition at slot c.

6 partitions:
#        size    offset     fstype [fsize bsize cpg/sgs]
 c:    472064     28672     unused      0     0        # (Cyl.     56 -    977)
 d:    500736         0     unused      0     0        # (Cyl.      0 -    977)
 e:     28640        32      MSDOS                     # (Cyl.      0*-     55)
 f:    472064     28672     4.2BSD      0     0     0  # (Cyl.     56 -    977)

Not updating disk label.
```

As we can see the disklabel is mostly complete, except that the NetBSD partition is assigned to partition ‘f’. It doesn't make sense to create several UFS partitions (in the disklabel sense) within the NetBSD MBR partition. In particular, note that we do not create any swap partitions, as swapping to a CF card will wear it very fast. Let’s change NetBSD partition letter to ‘a’, as the kernel likes it better that way:

```
# **disklabel -e wd1**
*[rename partition f->a]*
```

```
# disklabel wd1
[...]
6 partitions:
#        size    offset     fstype [fsize bsize cpg/sgs]
 a:    472064     28672     4.2BSD      0     0     0  # (Cyl.     56 -    977)
 c:    472064     28672     unused      0     0        # (Cyl.     56 -    977)
 d:    500736         0     unused      0     0        # (Cyl.      0 -    977)
 e:     28640        32      MSDOS                     # (Cyl.      0*-     55)
```

Now we format the two partitions we have just created:

```
# newfs_msdos wd1e
/dev/rwd1e: 28584 sectors in 3573 FAT12 clusters (4096 bytes/cluster)
MBR type: 1
bps=512 spc=8 res=1 nft=2 rde=512 sec=28640 mid=0xf8 spf=11 spt=32 hds=16 hid=32
# newfs wd1a
/dev/rwd1a: 230.5MB (472064 sectors) block size 8192, fragment size 1024
        using 6 cylinder groups of 38.42MB, 4918 blks, 9472 inodes.
super-block backups (for fsck -b #) at:
     [...]
```

Now the CF card is formatted, but is still completely empty. We will mount the freshly-formatted file systems and will start filling them with contents. In the examples below "..." stands for the directory with release sets (the directory you specified as an argument to the `-R` flag of the `build.sh` script if you did the build yourself). The examples below uses "hpcsh".

Important: make sure you mount the DOS file system with the `-l` option (see BUGS in mount_msdos(8)).

```
# mount -o softdep /dev/wd1a /mnt
# mount -o -l /dev/wd1e /mnt2
```

We will start with the DOS file system. As we said above, we want to put hpcboot.exe onto it:

```
# cp .../hpcsh/installation/hpcboot-sh3.exe /mnt2/hpcboot.exe
```

and the NetBSD kernel:

```
# tar -x -p -z -f .../hpcsh/binary/sets/kern-GENERIC.tgz -C /mnt2
```

The hpcboot.exe boot program can boot kernels from UFS, so you can skip this step, and extract the kernel to `/mnt` instead (where your NetBSD root partition of the CF card is mounted).

For now we are done with the DOS partition.

Next we will unpack the NetBSD release sets. The simple loop below extracts all the sets except the kernel ('k') and X11 ('x'). We already extracted the kernel to the DOS partition, so you always want to skip 'k*' sets. As for the X11, if you have a 512MB CF, you can as well extract them.

```
# for f in .../hpcsh/binary/sets/[^kx]*.tgz; do
>   tar -x -p -z -f $f -C /mnt
> done
```

With your CF now fully populated all that remains is some final touches to system configuration.

We create a mount point for the DOS partition:

```
# cd /mnt
# mkdir cf
```

And if you have put the kernel on the DOS partition (as we did in the example above), you create a symlink to the kernel. Strictly speaking, it's not necessary. NetBSD now uses ksyms(4), so programs that traditionally needed access to the NetBSD kernel image to parse its symbol table now use `/dev/ksyms`.

```
# ln -s cf/netbsd
```

Fix your localtime link:

```
# cd /mnt/etc
# rm localtime
# ln ../usr/share/zoneinfo/Europe/Moscow localtime
```

Edit fstab(5). We only have two partitions to add. Note that we mount the root file system with ‘noatime’ and ‘nodevmtime’ to reduce CF wear.

```
# cd /mnt/etc
# vi fstab
[...]
# cat fstab
/dev/wd0a       /       ffs     rw,noatime,nodevmtime   1 1
/dev/wd0e       /cf     msdos   -l,rw                   0 0
```

Edit rc.conf(5):

```
# cd /mnt/etc
# vi rc.conf
[...]
# sed -n '/^rc_configured/,$p' rc.conf
rc_configured=YES

# Add local overrides below
#

hostname="nada"

critical_filesystems_local="/cf $critical_filesystems_local"

no_swap=YES
savecore=NO
```

Here `/cf` is added to critical file systems so that `/netbsd` symlink works in case someone needs it, but as mentioned above, it’s not strictly necessary any more. Also, if you put the kernel to UFS, you don’t need that line either.

With `no_swap=YES` we tell that we intentionally configured the system without swap. Also, savecore is disabled as we don’t have `wd0b` anyway, so avoid complains.

Now check `/etc/ttys`, thought the defaults should be sane.

```
# cd /mnt/etc
# vi ttys
[...]
```

And finally populate `/dev`. If you are ok with 1000+ devices in `/dev`, you can just run:

```
# cd /mnt/dev
# sh MAKEDEV all
```

But if you want to avoid all those device nodes for raid and multiport serial cards you can spent just a little bit more time and create only those device nodes that you need. The example below is a baseline that is enough to give you a working system. Add more devices according to your needs.

```
# cd /mnt/dev
# sh MAKEDEV std
# sh MAKEDEV random systrace lkm clockctl
# sh MAKEDEV wscons
# sh MAKEDEV apm
# sh MAKEDEV scif0               # <- sh3 specific serial
# sh MAKEDEV ptm pty0 tty0 tty1
# sh MAKEDEV atabus0 atabus1 wd0 wd1
# sh MAKEDEV bpf0 bpf1
```

Congratulations! At this point your CF card is ready. You can plug it into you HPC and boot from it into multiuser.

Future Work

There are still a number of areas in which the handheld and pocket PC support within NetBSD can be further improved.

- The sound drivers need work – whilst we have VoIP and softphone capabilities within kphone and other packages, the sound drivers could do with an overhaul.
- Support for a windowing environment other than X would be beneficial. Qt/Embedded, Qtopia or GPE or a similar solution would be attractive to some PDA manufacturers
- Microsoft’s Windows environments have set the standard for integrated applications, such as Microsoft Office, on PDAs. Whilst Open Office or Star Office is probably too large to reside even on CF, and to run within acceptable limits on a PDA, it is still possible to use other open source applications for the same functions
- Wide-area communications would benefit from better 3G, GPRS, and soft modem support
- Java support in PDAs is becoming an increasing factor – whilst Sun’s JDK (versions 1.3 and 1.4) have been ported to i386 and SPARC platforms, there is little support for low-power CPUs and architectures in the standard JDK, although it is almost definitely too large and resource-hungry to fit on a PDA. Whilst there are many open-source Java Virtual machines in existence, such as Wonka, kaffe and sablevm, few have the Windowing Toolkit support to make them attractive propositions for devices with limited input options, and limited screen area. Sun’s J2ME is a possibility, more work needs to be done in this area.

The Ultimate Geek PDA

It is interesting to predict what the ultimate geek PDA will look like in a few years. We are seeing PDAs emerge from Japan right now with 2.1 inch screens, yet with VGA resolution. Much has emerged in this area over the last six months, and it is expected that this trend will continue. So it looks like the screen real-estate problem may be being addressed.

Input to the PDA remains a problem. It might be possible that voice recognition will gain ground in the next few years, rendering hunt-and-peck stylus input obsolete. Input is probably the most tortuous aspect to using a PDA these days.

Compact Flash cards may increase in size, and may also decrease in price, allowing the operating system to do more, and for applications to store more data.

Battery life may need to improve, and we may see more takeup, especially in Europe, of services more tightly integrated with what has been viewed as the telephone company domain. Indeed, this crossover into the realm of mobile telephone handsets has some interesting possibilities, and there is a lot of movement in this area.

The consolidation of a number of personal devices – pager, mobile telephone, PDA – into one PDA, Internet-enabled through 3G or GPRS, is an attractive possibility to a number of manufacturers, service providers and software vendors.

Conclusion

Handheld PCs and pocket PCs present a number of challenges for users – their size is what makes them appealing, and yet it is also the cause of some of the major human factors problems, most notably screen real-estate and keyboard and mouse input. Whilst NetBSD can be installed and will run very nicely on such machines, there is a certain amount of future work which needs to be done using NetBSD on these platforms to make the experience even better. These PDAs are not expected to be compute engines – their appeal lies in their portability and size. Cross-building of the operating system and the windowing system is obviously a necessary pre-requisite for developing with PDAs, and NetBSD has these features already. Cross-building of third-party applications for PDAs is also needed – this aspect has been addressed in other papers.