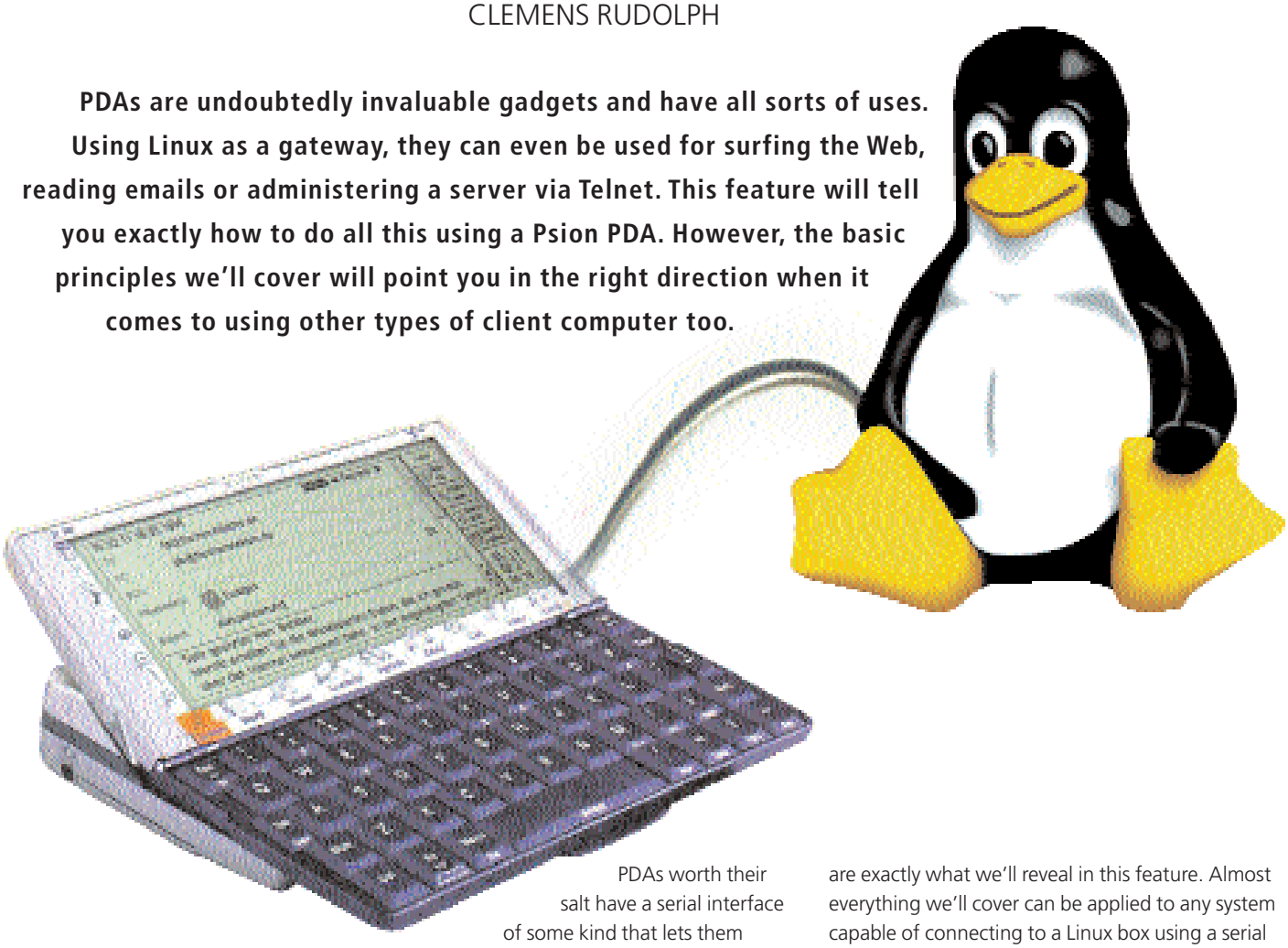
Turning PDAs into Linux terminals

BRAIN SWAP THE SECOND

CLEMENS RUDOLPH

PDAs are undoubtedly invaluable gadgets and have all sorts of uses. Using Linux as a gateway, they can even be used for surfing the Web, reading emails or administering a server via Telnet. This feature will tell you exactly how to do all this using a Psion PDA. However, the basic principles we'll cover will point you in the right direction when it comes to using other types of client computer too.



PDAs worth their salt have a serial interface of some kind that lets them communicate with the outside world. Most modern PDAs also include a range of Internet applications as standard, so all you need to get online is a modem. Alternatively, you can simply link them via a null modem cable to a Linux system that has an Internet connection. You'll need to know the secrets of *mgetty* and *pppd* in order to do so, but these (and how a Psion PDA can make use of them) are exactly what we'll reveal in this feature. Almost everything we'll cover can be applied to any system capable of connecting to a Linux box using a serial link, including old Atari, Amiga and even Intel 80286-based systems.

Preparing the Psion

Before we do anything else, we have to make the client computer – in our case the Psion organiser – Internet-capable. A glance in the Psion's system

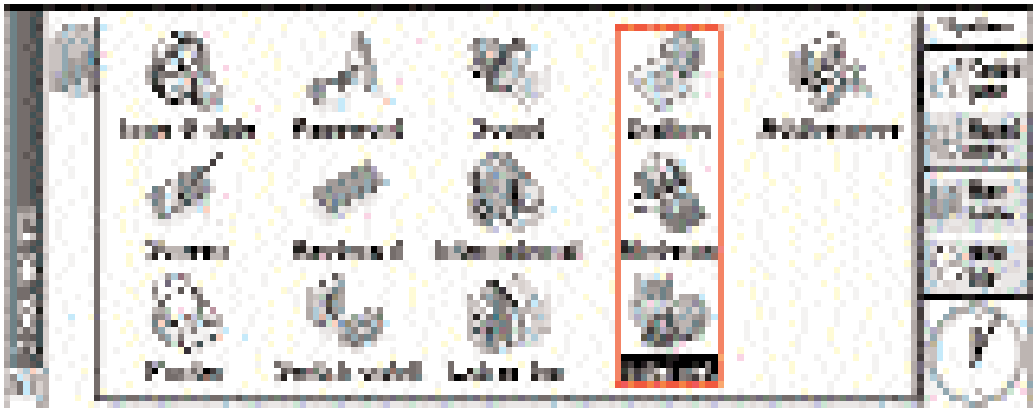


Fig. 1: Communication menus in the Psion's control panel

control panel (see Figure 1) reveals three icons used to configure connections with the outside world: Dialling, Modems and Internet. But since we'll be using a fixed connection, we'll ignore Dialling.

We'll start with the Modem configuration option. All we need to do is select "Direct connection" for the "Current modem" option. Next, turning to the Internet configuration option, we need to use the "New" option to make a new *profile* based on "Default settings", then give it an appropriate name – we'll use "Intranet", but anything will do. The settings that appear when you select the "Edit" option are shown in the screenshots in Figure 2. These are pretty straightforward to configure as needed.

At this point there is just one other step to complete on the Psion; as paradoxical as it may sound, in the PDA's main menu, under "Extras", we have to change the "Link" option to "OFF" in the "Communication" section. This is the only way to get the little organiser to perform purely TCP communication and to stop it acting as a PsiWin client. The next step is to start configuring our Linux box

Configuration of the gateway

Two basic types of connection are possible between a client and our Linux system. The first is a simple console connection, in which only a log-in (thus a console) is opened. The second type of connection makes use of the *Point-to-Point Protocol*, or PPP for short.

Using these two types of connection, the link between the Psion and the Internet can be made, the Psion's "Comms" terminal program making use of the console connection, and its "Web" and "Mail" programs making use of a "Point-to-Point" connection.

Port monitoring with mgetty

In order to build our communications framework, we first
n

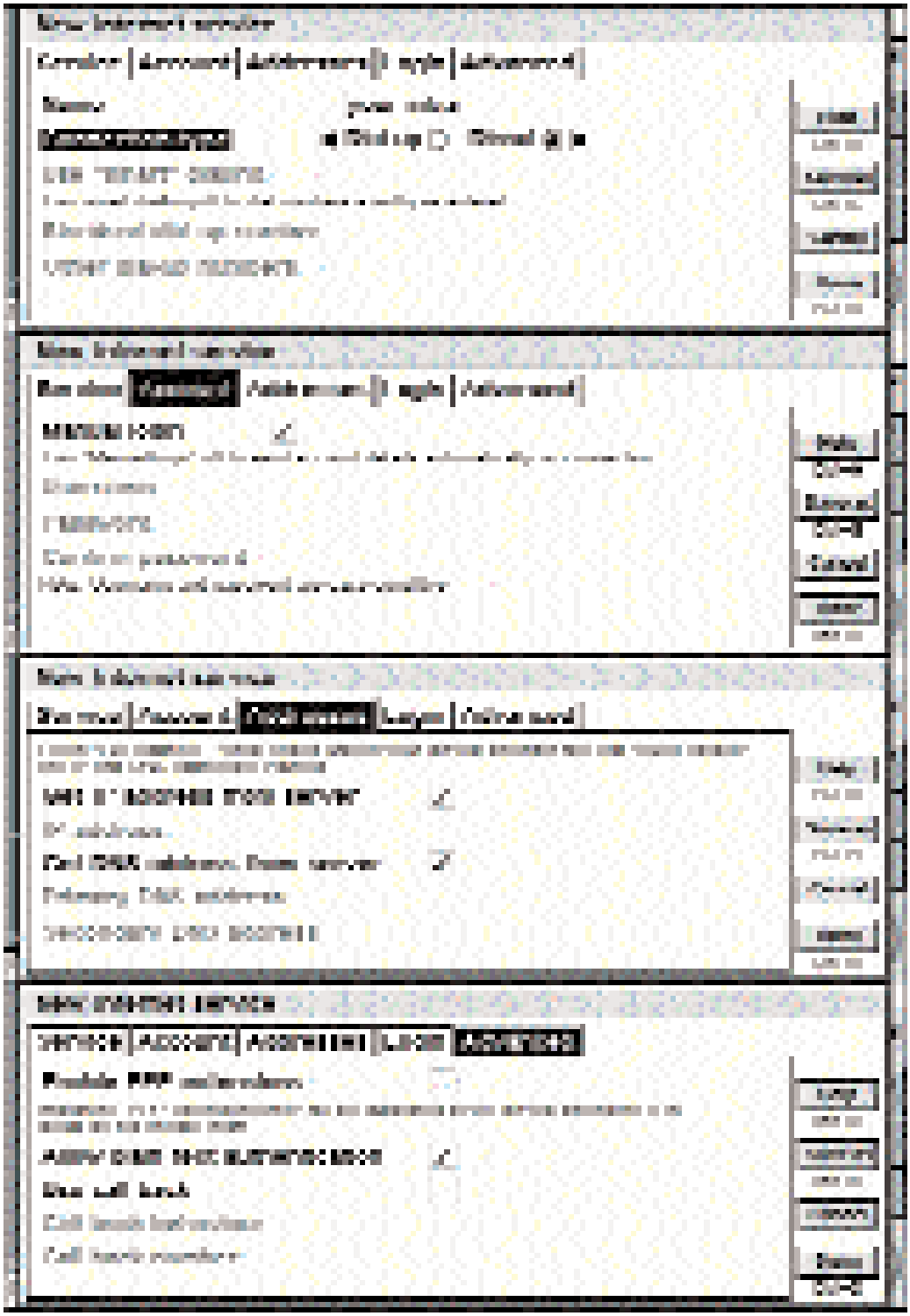


Fig. 2:
Configuring an Internet service

Espionage with logwin

When using a high debuglevel, as we have done in our examples, it makes sense to keep a close eye on the outputs of the services in order to see if anything is wrong. To do so, when working under X, you can use xterm (or similar) and an appropriate console to look at any interesting logfiles.
To achieve this for xterm (and its variants) we need to change to root mode using 'su -l', and then type

```
xterm -e tail -f /var/log/messages /var/log/warn /var/log/mgetty.ttyS[0-9]
```

The regular expression "[0-9]" at the end is necessary to capture only files ending in a number and not, for example, an old log file already automatically grabbed by the system. This would obviously cause some confusion in the case of tail.

Figure 3: Console log-in with Comms

Transport of configuration files

One option for transferring the configuration files required to give a Psion Internet access from one PC onto another would be to save the data to floppy disk. However, a much more elegant solution is to store the configuration onto the Psion itself and transfer it to the host as required. This does necessitate a program to be run on the host that allows a serial data transfer for files. This is where the almost ubiquitous minicom, which acts as a terminal emulator, comes in handy. By linking this with the Psion's Comms program it is very simple to exchange files between the two systems. The y-modem protocol comes in vary handy here. A few tips on how all this works can be found at http://www.mda.de/homes/tron5/psilink.html.

Table 1: The pppd-options and their effect

pppd-Option	Description
crtscts	This activates hardware flow control.
lock	This parameter serves to hang a UUCP-conform lock in front of the device in use. In other words an exclusive user right to the COM-port is established for the process accessing it at that moment (/var/lock/...).
noauth	With this option, authentication (which in our case would cause interference) is suppressed.
noccp	This option ensures that no CCP (Compression Control Protocol) negotiation occurs during the connection. The Linux machine and the Psion negotiate about compression at start of the connection process (whether wanted or not), but the pppd, which stems from a time in which telephone lines were even more unreliable than they are today, normally tries to keep negotiating every 10 seconds.
nopersist	This parameter is not absolutely necessary, as it applies by default. We have included it for completeness and security reasons though (see 'Read sequence of configs of pppd' in the text). In any case, it makes sure that the pppd stops when a serial connection (from the Psion) is closed. This, of course, only makes sense if an existing connection is checked by LCP (Line Control Protocol)
silent	Silent serves to ensure that the daemon continues to wait patiently until something stirs on the line. It instructs the daemon to wait, "silently" for whatever comes in and do nothing until then.
proxyarp	Using this parameter, an entry containing the IP address that we need for the Psion is added to the ARP table (or Address Resolution Protocol Table to give its proper name). Effectively, it means something like "It is now one of us".
local	This tells the daemon that it doesn't need to worry about a modem on the serial line, and so it

About the author

Clemens Rudolph is a programmer for an ISP, and specializes in PHP and Perl. He never tires of poking around in the bowels of his system.

"Comms versus Hermes"

Hermes, a Telnet client and terminal emulation program for the Psion (available from http://www.iota.demon.co.uk/psion/hermes/hermes.html), is a really amazing application that can provide a pure serial console connection as easily as a PPP-connection. In contrast, Comms is "only" a terminal and so can only be persuaded to work with the likes of login or minicom. This means that Hermes is really great when you need to log in to a Linux box on which you don't have root privileges. You can start pppd as normal user, provided the rights have been set accordingly and on call up the path is given as /usr/sbin/pppd .... You will need to make some configuration changes in Hermes in order to do so though. In the "Connection" menu, look for the "Connection" option and switch it to "TCP-Connection". Now all you need to do is enter the IP address of the destination system and you'll have a Telnet log-in in seconds.