Wipro UWIN

User Guide

Wipro Technologies
No.8, 7th main, I Block
Koramangala,
Bangalore – 560 034
India

uwin@wipro.com
http://www.wipro.com/uwin

CONTENTS

INTRODUCTION TO WIPRO UWIN 5

WHO NEEDS UWIN? 5
WHAT DOES UWIN PROVIDE TO A USER? 6
WHAT DOES UWIN PROVIDE TO A DEVELOPER? 6

UWIN INSTALLATION 7

MINIMUM SYSTEM REQUIREMENTS 7
INSTALLATION 7
UNINSTALLING UWIN 9

OVERVIEW OF UWIN 10

WHAT IS UWIN? 10
UWIN FEATURES 10
UNIX Shells 11
UWIN Case Sensitivity 11
Pathname mapping from UNIX to NT 12
UNIX Naming Convention 13
The File System 14
Converting Between POSIX and MS-DOS Text Format 15
File System Links 15
inet Daemon 16
Registry as filesystem 16
Working with Win32 Programs 17
Running CMD.EXE Built-In Commands 18
Stopping Win32 programs 18
Job Control 18
At and crontab 18
Aliases 19
VT100 Terminal Emulation 20
Process Control 20
File Descriptor semantics 20
UNIX Signal semantics 20
UCB Sockets based on WINSOCK 20
Support for UNIX devices 21
Working with tapes 22
Mapping to & from UWIN ids/permissions to Windows NT permissions 29
Memory mapping, shared memory and System V IPC 29
Error mapping from NT to UNIX 29
Internationalization in UWIN 29
UWIN Logs 30
Tracing 31
Invocation of UWIN commands from other applications. 32
Posix thread support in UWIN 32
Archiving using "Pax" 32
UWIN Control Panel Applet: 34
Conversion Utilities: 39

SERVICES AND DAEMONS 43

UWIN MASTER SERVICE (UMS) 43
Creation of /etc/passwd and /etc/group files 43
Starting of daemons 44
Setuid functionality 44
UWIN CLIENT SERVICE (UCS) 44
WORKING WITH REMOTE TOOLS 45

Using Telnet and in.telnetd *46*
ftp and in.ftpd *46*
rlogin and in.rlogind *46*
rsh and in.rshd *46*
rcp *46*
Configuration for rlogin/rsh/rcp *47*

TERMINAL EMULATION **51**

CUSTOMIZING THE COLOR AND SIZE OF THE TERMINAL 51
Enabling Scrolling *52*
Cut and Paste *52*
Title *52*
VT100 Attributes *53*
stty command *54*

UTILITIES AND APIS **57**

UTILITIES 57
keybind, vi_keybind, emacs_keybind *57*
popd, pushd & dirs *57*
ie *58*
title *58*
CONFIGURATION MANAGEMENT UTILITY 58
APIS 59

DIFFERENCES BETWEEN UNIX AND UWIN **61**

PATHNAMES ON UWIN 61
THE ADMINISTRATOR GROUP AND APPROPRIATE PRIVILEGES 61
ACCESSING REGISTRY THROUGH UWIN FILE SYSTEM 62
Finding UWIN Version information *62*

TROUBLESHOOTING **64**

Chapter 1
Introduction to Wipro UWIN

Wipro UWIN is a Unix to Windows Migration Toolkit that gives you all the features of a traditional UNIX operating system on Microsoft® Windows NT and Windows 95/98 platform. It enables UNIX applications to run on Windows with a mere recompilation. UWIN aims to build an Open Environment on Windows that will be both a good development environment and a suitable execution environment.

UWIN enables the user to use both Windows and UWIN ported Unix applications in the same desktop, without requiring a dual-boot.

This document provides installation information and a quick introduction to the UWIN product range.

Note: In future, all reference made to “UWIN” in this document refers to “Wipro UWIN”.

Who needs UWIN?

- Internet developers and ISPs who need tools and scripting (Apache, perl, shells and utilities)
- Major corporate IT shops that want to maintain a "write once - deploy many" policy for applications in the enterprise
- Corporate desktop sites that want to host both UNIX and Windows applications on a single platform
- System administrators who need remote login access using standard services, shells and utilities
- ISVs that want to use one workstation for developing both Windows and UNIX applications
- Developers building two & three tier client server applications
- Customers needing to port or develop background daemon applications, cron jobs and services
- Companies who want to capitalize on their UNIX asset investment as they move to Windows NT

What does UWIN provide to a User?

UWIN has a set of popular shells like ksh (Kornshell) & tcsh (C shell) and more than 300 utilities like vi, ls, ps, grep, tail, uudecode/uuendecode, mailx, find, perl, awk, etc along with a vt100 terminal emulation. It also provides a Telnet server along with other *inet* daemons and utilities like telnet, ftp, rsh, rlogin, and their corresponding servers for Windows NT, enabling a user to remotely access the system over the network. Optional tools include the Apache Webserver and *bind* DNS server.

What does UWIN provide to a Developer?

UWIN provides a UNIX API runtime library over the Microsoft Windows platform. It implements the set of UNIX APIs as a shared runtime library (DLL) using WIN32 APIs.

It provides a development environment for building UNIX applications on Windows. It has a cc compiler that is a wrapper around Microsoft Visual C++. UWIN also includes the GNU C++ compiler. It has the Windows runtime library for X11R6.4 to build and execute X11 applications on Windows. The curses library and vt100 terminal emulation are supported. Debugging can be done with any native Windows debugger. gdb, the GNU debugger can also be used for debugging applications developed using the gcc compiler.

Chapter 2
UWIN Installation

Minimum System Requirements

- Intel® x86, Pentium, Pentium Pro and compatible systems
- Microsoft® Windows® NT 4.0 or higher (Server or Workstation)/Microsoft Windows 95/98
- Microsoft Visual C/C++ 4.0 or higher (optional for development kits)
- 25-30MB of available hard-disk space (higher for FAT partition)
- Windows supported CD-ROM drive (optional]
- Winsock version 2.0

UWIN does not essentially require a NTFS partition. UWIN can be installed on a FAT partition as well. However, you will not get full POSIX features (e.g., hard-links, file-system security) on a FAT system.

Installation

UWIN can be easily installed from the self-extracting installation file, which is downloaded or from a CD-ROM.

You need to follow these steps before running the Installation program:

1. If this is not the first installation, Uwin Self extracting file will search for any previous installation of Uwin & will try to uninstall it. The un-installation procedure includes deletion of the Uwin installation directory. Take backup of all the data stored inside the Uwin directory.
2. It is not recommended to install UWIN on a remote share, which may be mounted as a Drive Letter.
3. For Windows NT, UWIN needs to be installed from the 'Administrator' user account, or from an account, which has Admin privileges. The installation will not proceed if this condition is not satisfied.
4. UWIN provides full POSIX file semantics only on NTFS partitions. It can be installed or used on a FAT partition. However, features like File links, file groups, ownership and the full scope of file names allowed under POSIX are available only when you are working on an NTFS partition. If you install the products on a

FAT file system, you will still get full POSIX file semantics for files stored on NTFS file systems.

To install UWIN, follow these steps:

1. Use File Manager, Explorer, or an MS-DOS Prompt, and run the Self-Extracting Uwin executable.
2. Follow the instructions during the installation
3. The default installation location is <SYSTEMDRIVE>:\Program Files\UWINX.X. The installation location is referred in the rest of the documentation as $UWIN_ROOT. You can change this install directory from the installation program
4. You can choose the UWIN components you wish to install in the installation program. The elements that are grayed are mandatory. It is recommended that you select all the components to prevent mismatch between versions due to incomplete installations
5. After the installation program copies the files, a Command prompt will pop-up and the UWIN part of the Installation will proceed. You will be asked to press *Enter* at the end of this phase of installation
6. At the end of the installation process, you will be prompted to view the README file. Please select this option, to start the HTML based README file. This file also contains the *Whats New* topics for the current release
7. Once the installation is complete, the Installation will ask you to reboot the system. After a reboot, UWIN will be ready for use.

Additional public domain and user contributed applications and utilities can be downloaded from the Wipro UWIN Website at http://www.wipro.com/uwin. Some of the utilities available at this website are:

- Apache 1.3.3, Perl 5.05, Lynx 2.8
- CVS 2.0, Tcl, VIM 5.3
- Flex, Rpcgen, atp 1.2b
- Tcsh, zlib library,
- Redhat package manager(rpm)
- X11R6.4 client libraries, Xterm

Uninstalling UWIN

To uninstall UWIN, you can either select the 'Uninstall' icon from the UWIN Program Group created during installation or select the *Wipro UWIN X.X* item from *"Add/Remove Program"* applet of the Control Panel.

Note: Uninstall will delete all the contents under the Uwin directory structure; please take backup of the user created files stored under Uwin directory .

As soon as the un-installation process begins, a command prompt will pop up which will run a script for killing all Uwin related process. This script runs for few seconds & will be terminated automatically. From then on the deletion of files, which are logged for un-installation starts. A status bar will appear which indicates the user about the status of progress of un-installation.

After all the files logged for un-installation is deleted, a small program will be executed that takes care of removing all the remaining uwin related data from the installed system.

If the un-installation is able to stop all the currently running processes, needed for un-installation, then it will complete the un-installation without asking for a restart. Otherwise the system will ask the user to reboot the system so as to stop all the processes, related to un-installation, & make the system clean.

Chapter 3
Overview of UWIN

What is UWIN?

The UWIN environment is a set of shared libraries (DLLs) that is built over the Win32 subsystem of Windows. UWIN applications work as a user-level native Windows process and uses the UWIN shared libraries. This approach allows UWIN to work equally well on Windows NT and Windows 9x platforms.

UWIN Features

UWIN provides all features dictated by the POSIX.1 and POSIX.2 standards derived from the UNIX operating system. Some of the important features are:

Unix shells
Pathname mapping from Unix to Windows
Unix Naming Convention
File System
POSIX and MS-DOS text format conversion
File system links
inet Daemon
Registry as filesystem
Working with Win32 Programs
Job Control
Shell Aliases
vt100 Terminal Emulation

Process control and management
File descriptor semantics
Unix signal semantics

UCB sockets based on Winsock
Support for Unix devices
Mapping to and from UNIX Ids or permissions to NT
Memory Mapping, Shared Memory and System V IPC
Error Mapping from Windows to Unix

UNIX Shells

Users may use KornShell93 (ksh), Bourne Shell (bash) or the C Shell (tcsh). The shells support all the usual features, including job control, command pipelines, and #! script interpretation.

Many people prefer the C shell as their interactive shell, but use the Bourne shell or the KornShell as their script processor. The KornShell is a superset of the traditional UNIX Bourne shell, and conforms to the requirements for a POSIX.2 shell.

UWIN Case Sensitivity

By default, UWIN will be case insensitive. But, it is possible to maintain the case sensitivity in Windows using UWIN, by doing the following configuration settings:

1. Click Start->Settings->"Control Panel".
2. Double click on "Wipro UWIN".
3. Click on the "System" tab.
4. Select the "Category" "Miscellaneous".
5. Check the box "File system case sensitive".
6. Click on "Apply" and "OK".
7. Restart UWIN.

Note: Restarting UWIN could be done in two ways.

1. Hard reboot.

Restart the Windows.

2. Soft reboot.

Kill all UWIN processes by executing the following command from the UWIN Window:

```
$ /etc/stop_uwin.sh
```

Once all the UWIN processes are killed (you can view your task manager for the presence of init.exe, inetd.exe, at.svc, ksh.exe

etc), open a new MS-DOS window and execute the following command:

C:\> net start uwin_ms

Now if you open any new UWIN Window, it would preserve the case.

Pathname mapping from UNIX to NT

UWIN provides UNIX style naming for all files with / as the filename delimiter. / is mounted on to the directory where UWIN is installed, which can be selected at the time of UWIN Installation. UWIN provides a mountable filesystem view of the underlying Windows filesystem.

If UWIN is installed in the directory C:\Program Files\UWIN\, then / will refer to C:\Program Files\UWIN\. A pathname that begins with / is hence considered relative to the UWIN Installation Directory.

Certain directories are implicitly mounted at the time of system reboot by UWIN. /sys is mounted as the Windows System Directory, which typically refers to C:\WINNT\System32 for Windows NT systems and C:\Windows\System for Win9x systems.

/msdev is mounted as the Microsoft Development Kit directory, which is typically C:\Program Files\DevStudio\ for Microsoft Visual C++ 5.0 and C:\Program Files\Microsoft Visual Studio\vc98\ for Microsoft Visual C++ 6.0.

The Unix character and block special devices are located in /dev.

Each drive letter in Windows is mounted in a single letter directory name under /. C: Drive can be accessed as /C or //C and D: Drive can be accessed as /D or //D, and so on for all the available drives. Hence, /C/file.c or //C/file.c is the UWIN name for C:\file.c.

Following are the default mounts in UWIN:

Mount Point	Equivalent Windows Directory
/	UWIN Install Directory
/C	C: Drive (similarly for other Drives)
/sys	Windows System Directory
/msdev	Microsoft Development Kit/Compiler Directory
/reg	Windows Registry
/win	Windows system directory (eg. D:\winnt

/proc	/usr/proc – directory containing information on the running processes

The "Administrator" can create new mount using the mount command. Additionally, the Administrator can specify the mount points in /etc/fstab, which are auto-mounted at system boot. By default, the directories are mounted without case distinction. However, the mount command allows directories to be mounted as case sensitive so that the files makefile and Makefile are different. Please see the manpage of mount for usage information.

winpath

You can get the equivalent Windows path for any given Unix path using the /bin/winpath command.

"winpath" can be used to convert UNIX path to WIN32 pathnames. This is equivalent to **-H** option of typeset. winpath is inverse of unixpath command.

Options:

-q, --quote Quote the Win32 pathname if necessary so that it can be used by the shell.

Examples:

$ winpath /e/uwin
e:\uwin
$ winpath /sys
d:\winnt\system32

$ winpath /tmp/uwin_log
D:\Program Files\UWIN2.9/tmp\uwin_log
$
$ winpath /d/notes.txt
D:\notes.txt
$

UNIX Naming Convention

UWIN allows UNC naming conventions found on Windows systems.

Ex: \\machine-name\share-name\file can be accessed as
//machine name/share-name/file in UWIN.

The UWIN naming convention is same as that in UNIX. The character set for filenames on UWIN is the set specified by POSIX: All printable characters are allowed in filenames except for
/, \ ,:, *, ? , “, |, ', &, <, >, ;, and ,

The File System

For the closest UNIX-like behavior, you should install and use UWIN on a NTFS file system. NTFS supports the following requirements of a POSIX file system.

Pathnames use / as the pathname separator.

Pathnames starting with //D refer to files on a particular drive. For example, pathnames beginning with //C is on the C: drive.

Filenames are case preserving. Files created with name “Temp”, are stored and displayed as “Temp”, but you cannot create another file with the same name and different case.

On a FAT file system or across a network, file names are case-insensitive, though case is generally preserved. For reasons internal to Windows NT, file specifications using shell wildcards (such as ? or *) are case-sensitive on a FAT filesystem and across a network.

NTFS filenames allow the full character set specified by POSIX.1. The following characters are not allowed in filenames:

\ / : * ? ! | " < > Control characters

The following points worth noting

- Hard links is supported (files can have more than one name).
- Real user and group names appear as the file owner and file group when you do a long directory listing. (Note: Files can be owned by a group unlike the traditional UNIX implementation.)

On a FAT file system, all files are owned by the user "Everyone" and the group "Everyone". You can determine the type of a file system by using the df command on a directory in that file system.

If you are using both UWIN utilities and Win32 applications on the same file, you need to remember the following:

- Win32 applications do not provide access to case-sensitive file-names or to hard links.
- Most UWIN utilities expect lines in shell script files to end in a line-feed only. The UWIN ed, ex, and vi editors maintain the files as found, but some Win32 editors do not.

Working with Samba file system

It is possible to mount a samba file system on Windows using UWIN. This could be accomplished by following the steps given below:

1. Install Samba on your Linux/Unix system.
2. Double click on "Network Neighborhood" on your Windows machine.
3. Map any directory in your samba system as a drive on Windows.
4. Start a new UWIN window.

With this configuration, you can traverse to any directory in the Linux/Unix system, provided you have the proper access rights.

Converting Between POSIX and MS-DOS Text Format

Win32 applications use the MS-DOS text format. A *carriage-return/line-feed* marks the end-of-line in a text file. POSIX systems use the POSIX text format, where end-of-line is marked only by the character *line-feed*. Most of the UWIN utilities require their scripts to be in POSIX format.

The nocrnl utility converts carriage-return/line-feeds to line-feeds in text files.

```
$ nocrnl myscript.sh
```

will convert myscript.sh from MS-DOS format to POSIX format.

Conversion from POSIX to MS-DOS format can be done using "-D" option of the "cat" command.

E.g.:

```
$ cat –D myscript.sh
```

The above command prints the output on the standard output, so you need to redirect the output to a file using "tee" or ">".

Eg:

```
$ cat –D myscript.sh > myscript1.sh
```

or

```
$ cat –D myscript.sh | tee myscript1.sh
```

File System Links

A link is a name for a file. Symbolic links to files and directories can be created and recognized by all programs linked with the UWIN

library. Shortcuts created by Windows 9x/NT appear as symbolic links under UWIN.

On NTFS partition, files may have more than one link. Alternate names using links have the advantage of being more like the traditional UNIX environment.

For example, with the correct PATH these commands enable you to run the program TELNET.EXE with the command Win-telnet:

```
$ cd /sys
$ ln TELNET.EXE Win-telnet
```

Links can be useful if many users have accounts on one UWIN machine. Links to Win32 commands can be put in a directory already in the users' default PATH, so no users need to change their environments.

inet Daemon

inetd, the Internet super server invokes all the other daemons like the in.telnetd, in.ftpd, in.rlogind, and in.rshd. The telnet daemon allows users to logon to the NT system from the network. The ftp daemon allows users to transfer data over the network. The rlogin daemon allows users to login from a remote machine. The rsh daemon allows users to execute a command from a remote machine.

The UWIN Master Service (UMS) is a Windows NT based service, which starts at system boot. On startup, it runs the /etc/rc script. inetd is started through this script. Based on the entries in /etc/inetd.conf, inetd waits for incoming connection, and starts the corresponding daemon when a remote network request is received.

Registry as filesystem

The Windows Registry is treated as a file system in UWIN, with Registry Keys treated as files and keys that have sub-keys treated as directories. The Values associated with the Keys and their Data are considered as files, which can be edited by any UWIN editor (e.g. vi, pico, vim).

The tools and utilities in the UWIN Toolkit can be directly used on the registry keys. The registry is automatically mounted under /reg during startup. The values are represented in hexadecimal values in these files.

Working with Win32 Programs

Invoking Win32 Programs

UWIN allows you to run Win32 programs from UWIN shell prompt. But when running a Win32 program from the UWIN environment, keep in mind the distinction between the name of the program to be run and the arguments to the program. The name must be in POSIX format; the arguments must be in the format expected by the program (usually Win32 format).

Ex: Notepad can be invoked with the command

```
$ /sys/notepad
```

assuming the Win32 programs are stored in /sys and that the Notepad program file is named notepad.exe. If Notepad is started to edit the MS-DOS file /C/readme.txt, the command is

```
$ /sys/notepad.exe C:\\readme.txt
```

Note that the argument, the pathname of the file readme.txt, is in the Win32 format. The double \ is necessary because the UWIN shells treat backslash as a special character, hence needs to be escaped. The filename could also be surrounded in single quotes, like this:

```
$ /sys/notepad.exe “C:\readme.txt”
```

These techniques have some inconveniences:

Control isn't returned to your terminal until after the Win32 program has exited. This can be solved by running the command in the background using the & metacharacter:

```
$ /sys/notepad.exe &
$ /sys/notepad.exe C:\\readme.txt &
```

It's cumbersome to type the entire pathname to the Win32 program. One of the following options can be adopted to get around this problem:

- You can add the Win32 commands to the environment by adding the appropriate directories to the PATH environment variable. For example, if the Win32 commands are in the directory /sys, the following line can be added to the /etc/profile file:
  export PATH=$PATH:/sys:

The pathname is added in the POSIX filename format, not the Win32 format. All of the program files in the Windows NT system32 directory are available from the UWIN command line. Invoking Notepad is now just a matter of typing:

```
notepad.exe
```

This is the most general solution, but one still has to remember file extensions when running the command.

- Creation of aliases in the environment that refer to Win32 programs.
- Creation of one alias for each program.
- Creation of hard links to the Win32 programs, effectively making "copies" with more convenient names. This only works if the Win32 programs are installed on an NTFS drive.

Running CMD.EXE Built-In Commands

The Windows NT command processor CMD.EXE can be run from ksh, so CMD.EXE built-in commands can also be run. Assuming that PATH has been set up to include the Windows NT system directory, UWIN environment could be set to give the dir command for directory listings:

```
$ alias dir='CMD.EXE /c "dir /on"'
```

Now, the dir command can be used in Ksh, and it will produce MS-DOS style listing.

Stopping Win32 programs

Apart from UWIN applications, the kill command can stop a Win32 process as well. UWIN command should be used to terminate UWIN applications. When a UWIN process is killed from the Win32 environment, certain important clean up functions may not be performed, resulting in a possible memory leakage.

Job Control

The UWIN command shells support *job control*. A *job* can be suspended with Control-Z and restarted in the foreground using fg command or in the background using the bg command or killed using the kill command. All the jobs running in the background can be displayed with the jobs command.

At and crontab

“at” and “crontab” in UWIN works similar to its counterparts in UNIX.

This following section gives you some configuration checks for at and cron functioning.

Check points:

1. ums and inetd should be running.
   Check from the NT task manager for the following processes

ums.exe
inetd.exe
at.svc (2 instances)

If the above processes are not in the processes list, you need to start UMS using the following command from ksh or MS-DOS prompt

$ net start uwin_ms

2. ucs should be installed for the user account from which you schedule jobs. You can install ucs for the user by telnet-ing to the system with the same user id and corresponding password

   For e.g.: if you want to schedule jobs with the user id "uwin", You telnet to the host using user id "uwin" and its passwd

3. Then you can use one of the following to schedule the job.

Just to check if you can schedule a job using "at" using the following command on the ksh prompt

$ at now pwd > pwd.log <Enter>
^D

Aliases

An alias is a command-word that replaces a long command. For example, Ksh has the command r (for repeat) which is actually the command *fc -s*; when you type r as a command the Ksh substitute fc -s. This makes aliases an excellent way to eliminate the need to type the extension .exe at the end of a command name.

Aliases are ideal for applications such as word processors, where the directory containing the application may have only one executable file. Adding the directory to the PATH clutters the environment variable to no real purpose. Anyone can set up aliases, even without Administrator privileges.

To create an alias for Microsoft Word in the Ksh :

$ alias word="/C/Program Files/winword/WINWORD.EXE"

Instead of executing the frequently used aliases for each shell, it is advised to include them in /etc/profile.

VT100 Terminal Emulation

The POSIX termios interface is supported for consoles with vt100 terminal emulation. The terminal is the console window displayed when a UWIN shell is run. It is possible to resize the terminal, change the colors and scroll back through the previously displayed material. The properties of the terminal window can be controlled both through the stty command and by adjusting the Windows properties of the console window.

UWIN supports vt100 terminal emulation in its consoles.

Process Control

With the exec family of functions, an existing process can be overlaid with another process. Each process has a unique process-id and each process belongs to a process group. While processes can be created using the fork() function, the vfork() function is also an efficient way to create processes. UWIN also includes a spawn family of functions that combines the functionality of fork/exec for efficiency.

File Descriptor semantics

Open files, pipes, sockets, fifos and character and block special device files have file descriptors associated with them. They can be duplicated and inherited with UNIX semantics.

UNIX Signal semantics

Nearly all of the UNIX signals are provided including job control signals so that ksh can stop and restart jobs. A process can catch, block or ignore signals. Signals can be sent to process or to process groups. Applications compiled with the –D_BSDCOMPAT flag obey UCB UNIX signal semantics.

UCB Sockets based on WINSOCK

The socket interface uses the UCB header files and naming conventions, but is implemented as calls to WINSOCK. Sockets are file descriptors and obey file descriptor semantics. Both Internet domain protocol (i.e. AF_INET family) and UNIX domain sockets are provided. The multicast socket protocol is supported. The connect stream library, which provides a higher level and easier to user interface, is also provided.

Support for UNIX devices

UWIN provides character and block devices with major and minor numbers as found on UNIX systems. Support for direct access to floppy drives (/dev/fd0) and SCSI tape drives (/dev/mt0), as well as ptys and ttys is available.

Virtual Devices:

A number of devices in the UWIN subsystem are virtual devices. They do not have an implementation in the file system. The /dev/null device, the ttys (including /dev/tty) and the pseudo terminals, serial lines and sockets have been designated as virtual terminals.

Like traditional UNIX systems and unlike the Win32 environment, the null device is named /dev/null (under Win32, the null device is named NULL). You can redirect input and output to and from /dev/null.

Terminal devices also exist and can be listed with the tty command. The controlling terminal /dev/tty exists, as do terminals named /dev/ttyxx, where x is a digit. The POSIX termios interface is supported for consoles with vt100 emulation.

The pseudo terminals are used by different utilities, including xterm and telnet. UWIN allows 128 pseudoterminals. A pseudoterminal being used by xterm is not available for use in a telnet session.

The command

```
ls /dev
```

lists the contents of the virtual device space.

The virtual devices are:

Device Name	No. of devices	Device Name
Floppy Devices	2	/dev/fd[0-1]
Parallel port	2	/dev/lp[0-1]
Serial Port	8	/dev/mod[0-8]
SCSI Tape Drive	8	/dev/rmt[0-7] (Rewind on close) /dev/rmt[0-7]n (No Rewind on close)
Pseudo Terminals	128	/dev/pty[p0-wf] (Master side) /dev/tty[p0-wf] (Slave side)
Console Terminals	246	/dev/tty[10-255]

Serial Terminal Interface	10	/dev/tty[00-09]
Std Handles	3	/dev/stdin (Standard input) /dev/stdout (Standard output) /dev/stderr (Standard error)
Special Devices	2	/dev/tty (Standard terminal) /dev/ptymx (Pseudo terminal) /dev/null (Null Device)
Windows clipboard	1	/dev/clipboard (Device to access Windows clipboard)

Working with tapes

This section describes the tapes implementation in UWIN. The various tape functions, which enable applications to read from, write to, initialize, control, and retrieve tape information are discussed in detail.

Introduction

You can write into the tape using standard tools like tar, dd, pax and cpio etc. You can read from tape using tools like dd, pax etc.

You can control the tape using ioctl() calls; send commands to the tape drive using tools like mt.

The tape is a block device and data transfer happens in blocks of (default) 512 bytes. Of course, this blocksize can be set to a multiple of 512 by the user. There are some additional properties, which the user can set in the tape device using the minor number to store the info. (See Layout of minor device byte for more detail)

Addressing devices

There will be four tape devices that will come pre-configured with the UWIN package (*see the table below for details*). However, the user may decide to configure the tape device using *mknod (*you need to be an administrator to run *mknod*) and specify any device name he wants. The tape device like any UNIX device, is identified by its major number and minor number. *The major number of tape devices in UWIN is 1* . The minor number of the tape stores information such as "***BSD behavior***", "***rewind-on-close***", etc. These properties are specified in the header file <sys/mtio.h> . The user has the option of setting these options in the tape device. Tape devices have been implemented as fixed block I/O devices having a

fixed block size that can be configured. Read and write are expected to happen on multiples of this block size only.

Pre-configured tape devices in UWIN

UWIN Name	Windows name	Rewind-on-close	BSD behavior
/dev/mt0	\\.\Tape0	Yes	No
/dev/mt0n	\\.\Tape0	No	No
/dev/mt1	\\.\Tape1	Yes	No
/dev/mt1n	\\.\Tape1	No	No

From the UWIN tape device name, you can also find the actual Win32 pathname (e.g. \\.\tape0) using the uwin_path().

Layout of minor device byte

NRWND	BSD	0	0	0	T2	T1	T0	

NRWND:7 BSD:6 RESERVED:5-3 X:2-0]

Bits 0-2 : (T0,T1,T2) These bits make up the 'X' of the physical Windows path (*\\.\tapeX*) of the tape drive
Bit 7 : NRWND: No rewind on close
Bit 6 : BSD : BSD behavior

So if you have two tape drives and you want to configure the tape drive with the higher id as follows

No BSD behavior
no-rewind-on-close
name is /dev/st1

Then you would issue the following command mknod -m 0666 /dev/st1 b 1 129

Note: If your system has only a single tape drive, then the system device identifier will be tape0, and the physical definition will be \\.\tape0. So, in this case, the bits T2, T1, T0 would be 0,0,0. If you have more than one tape drive, then their physical definition would be automatically assigned by Windows (NTDETECT.COM in NT) at startup in the increasing order of their SCSI id's. So if you have two tape drives with SCSI id's 3 and 5, then the tape drive with SCSI id 3 would be \\.\tape0 and the tape drive with SCSI id 5 would be \\.\tape1.

Opening a tape

You use the open() function to open the tape device. The open on a tape device can fail if a tape is not inserted, resulting in the error report EIO. An open can also fail if the tape controller associated with the special file is not detected by the driver. In this case, the error reported is ENXIO.

If the tape is opened for reading and a no-rewind device has been specified, the close advances the tape past the next filemark (unless the current file position is at EOM) leaving the tape correctly positioned to read the first record of the next file. However, if the tape is at the first record of a file it doesn't advance again to the first record of the next file. These semantics are different from the older BSD behavior. If BSD behavior is specified then no implicit space operation is executed on close.

Reading

The standard read() system call shall be used to read from the tape. A read reads the next record on the tape. The record size is passed back as the number of bytes read, provided it is no greater than the number requested. When a tape mark or end of data is read, a zero byte count is returned and another read will return an error. This is different from the older BSD behavior where another read will fetch the first record of the next file in the tape. If this behavior is required, the device should be configured as specified in the layout of the minor byte. Two successive reads returning zero byte counts indicate the EOM. No further reading should be performed past the EOM. The UWIN tape device is a fixed-length I/O device and it requires the number of bytes read to be a multiple of the physical block size. Read requests that are lesser than a physical tape record are not allowed and error invalid argument (EINVAL) is reported.

Writing

The standard write is used to write to the tape. A write writes the next record on the tape. The number of bytes written needs to be a multiple of the physical blocksize (a multiple of 512). When logical EOT (End Of Tape) is encountered, the number of bytes that have been written will be returned. Another write will return a zero byte count. The next write will receive an error code of ENOSPC.

If data was written, a file mark is automatically written by the driver upon close. If the rewinding device was specified, then upon close the tape will be rewound after the file mark is written. If the user wrote a file mark prior to closing, then no file mark is written upon close. If a file positioning ioctl, like rewind, is issued after writing, a file mark is written before repositioning the tape.

Controlling a tape

There are standard unix tools like **tapecntl** or **mt** to issue various commands and control the tape. These programs can be used to control tapes in UWIN. The *ioctl* system call is used to control the tape device. The following **MTIOCTOP** (Magnetic Tape Input Output Control OPeration) operations are supported in UWIN.

MTWEOF: Write an end-of-file record

MTFSF: Forward space over file mark(i.e. position at first record of next file)

When spacing forward over file marks (EOF records), the tape head is positioned in the tape gap between the next EOF record and the record that follows it.

MTFSR: Forward space to inter-record gap

When spacing forward over a record (either data or EOF), the tape head is positioned in the tape gap between the record just skipped and the next record. Record skipping does not go past a file mark; file skipping does not go past the EOM; After an MTFSR <huge number> command the driver leaves the tape logically positioned before the EOF. A related feature is that EOFs remain pending until the tape is closed. For example, a program which first reads all the records of a file up to and including the EOF and then performs an MTFSF command will leave the tape positioned just after that same EOF, rather than skipping the next file.

MTBSF: Backward space over file mark

When spacing backward over file marks (EOF records), the tape head is positioned in the tape gap preceding the EOF. Thus the next read would fetch the EOF.

MTBSR: Backward space to inter-record gap

When spacing backward over a record (either data or EOF), the tape head is positioned in the tape gap immediately preceding the tape record where the tape head is currently positioned.

MTREW: Rewind
MTOFFL: Rewind and take the drive off-line (i.e. eject)

MTOFFL rewinds and, if appropriate, takes the device offline by unloading the tape. The tape must be inserted before the tape device can be used again.

MTNOP: No operation, sets status only (no op, set status only (read with MTIOCGET))
MTRETEN: Retension the tape

The MTRETEN retension command is used to restore tape tension, improving the tape's soft error rate after extensive start-stop operations or long-term storage.

MTERASE: Erase the entire tape and rewind

MTERASE rewinds the tape, erases it completely, and returns to the beginning of tape.

MTEOM: Position to EOM (goto end of recorded media i.e. after last file mark (for appending files))

MTEOM positions the tape at a location just after the last file written on the tape. This is after the last file mark on the tape. Additional files can then be appended onto the tape from that point.

MTSRSZ or **MTSETBLK:** Set record size (set block length)

MTGRSZ: Get record size
MTSRSZ and MTGRSZ are used to set and get fixed record lengths. The MTSRSZ command allows a user-defined blocksize to be set (though it must be a multiple of the physical block size of the tape device).

MTSEEK: Seek to block

Seek though ioctl() is not implemented in UWIN though the lseek() is allowed and has been implemented.

Tape Properties

BSD Behavior

The following constitute BSD behavior:

No implicit space operation is executed on close. *So if BSD behavior is specified then the no-rewind-on-close flag is ignored*.

When a tape mark or end of data is read, a zero byte count is returned; another read will fetch the first record of the next file in the tape.

Rewind on Close

As the name suggests, the tape will be rewound when a close is issued. A read from the tape after a subsequent open will read the first record of the first file. If, however, the tape is a no-rewind-on-close device then the close advances the tape past the next filemark (unless the current file position is at EOM) leaving the tape correctly positioned to read the first record of the next file. However, if the tape is at the first record of a file it doesn't advance again to the first record of the next file.

Examples

[Example] 1]

Assume that we have a single tape device in our system. We can have the same physical device display different properties. In our case lets have the following two possibilities.

Case(i)

Device properties: Name: "/dev/mt0"; Rewind on close; No BSD behavior.

This device has been configured thus in UWIN.

```
mknod -m 0666 /dev/mt0 b 1 0
```

Case(ii)

Device properties: Name: "/dev/mt0n"; No BSD behavior; No rewind on close.

This device has been configured thus in UWIN.

```
mknod -m 0666 /dev/mt0n b 1 128
```

Suppose you have three files on the tape.

```
$ dd if=/dev/mt0
```

[*This will read the first file on the tape and write it on the standard output]*

$ dd if=/dev/mt0
[*This will also read the same file i.e. the first one on the tape and write it on the standard output.*
This behavior happens because /dev/mt0 is a rewind-on-close device]

$ dd if=/dev/mt0n
[*This will read the first file on the tape and write it on the standard output*]

$ dd if=/dev/mt0n
[*This will read the second file on the tape and write it on the standard output. This is because /dev/mt0n is configured as a no-rewind-on-close device*]

[**Example 2**]
A small program which will get the record size information from the tape and then eject the tape from the drive to put it offline.

```
#include <stdio.h>
#include <sys/mtio.h>
#include <fcntl.h>
#include <errno.h>

int main()
{
        int fd;
    struct mtop s;

    if (( fd = open("/dev/mt0",O_RDONLY)) < 0)
     {
        perror("open failed : %d\n");
        exit(1);
     }
    s.mt_op = MTGRSZ;  /* Get Record Size */
    if (ioctl(fd, MTIOCTOP, &s) == -1)
     {
       perror("ioctl failed");
       exit(1);
     }
     else
         printf("Blocksize = s.mt_count = %d\n",s.mt_count);
```

```
    s.mt_op = MTOFFL;    /* Put the drive offline */
    if (ioctl(fd, MTIOCTOP, &s) == -1)
    {
      perror("ioctl failed");
       exit(1);
    }
}
/* Note:  The same can be achieved using the GNU mt tool. */
```

[**Example 3**]

```
$ pax -wvf /dev/mt0 /usr/man
[This will create a pax archive on the tape ]

$ tar -tvf /dev/mt0
[This will list the details of the information in the tape in a tabular
form ]

$ dd if=/dev/mt0 of=/tmp/output_file
[This will copy the pax archive read from the tape into the
output_file in the /tmp directory]

$ cat < /dev/mt0 > /tmp/output_file
[This achieves the same as the above ]
```

Mapping to & from UWIN ids/permissions to Windows NT permissions

Windows NT subject identifiers are mapped to UWIN user ids and group ids. UWIN permissions are mapped onto Windows NT ACLs. The Administrator can use chown() to change the owner and/or group of a file. Setuid functionality is also available in UWIN.

Memory mapping, shared memory and System V IPC

Both mmap() and the System V shared memory facilities are provided. The System V semaphore, fifos (UNIX named pipes) and message queues are implemented.

Error mapping from NT to UNIX

Errors returned by WIN32 functions are mapped into UNIX errnos.

Internationalization in UWIN

Wipro UWIN 3.0 onwards includes internationalization support. Support for Wide character IO and regular expressions are also added.

UWIN Logs

UWIN has several error logging mechanisms. The following are the list of log files associated with UWIN processes and services.

Log file	Description
/tmp/install_log	Proceedings/problems with UWIN Installation
tmp/uwin_log	Unix APIs and utilities' log
/tmp/ums.out	WIN Master Service logs
/tmp/ucs.out	UWIN Client Service logs
/usr/lib/at/log	UWIN "at" service logs

install_log : This is created during the installation of UWIN. It gives a brief on the errors that occurred during the UWIN setup, where the script sets permissions and links. This can be referred if you face problems after a fresh installation/upgradation.

uwin_log : This is a very important log which gives you logs from the utilities, daemons, and other programs which accesses UWIN dlls. It gives information like date, time, dll/exe name, function called, error description and error number. This is very vital during debugging of the application you have written/ported on to UWIN.

ums.out : This log is used by ums to log its proceedings and errors. This gives you information on passwd file generation when mkpasswd is executed. Various daemons logs its erros and connection information when remote tools like telnet, ftp, rsh, rlogin are used. Setuid errors can also be found here.

ucs.out : This log provides information on ucs service installation * startup.

/usr/lib/at/log : This log gives provides the status on the jobs scheduled. It also logs any errors occurred during the execution of the jobs.

Note: uwin_log gets copied onto uwin_log.old with system/UWIN restart. ums.out gets reset when ums is restarted.

Tracing

UWIN provides commands to trace the system calls and daemon processes.

Trace a command:

/bin/trace.exe can be used to trace the system calls that a command makes. This can also be used to trace a custom executable you have built with UWIN.

Usage:

$ trace –it <command>

You can also get the trace output into a file using the "-o" option.

$ trace –it ls –o /tmp/ls.log

Trace can also be used when an application/utility hangs or misbehaves to track down to the exact API call, which is failing.

traceit

traceit is used to trace daemon processes. Daemons like in.telnetd, in.ftpd, in.rshd can be traced using this command.

Usage: traceit daemon

Example:

$ /etc/traceit /etc/in.telnetd

The trace log will be created in /tmp/log/<daemon name>, for in.telnetd it will be /tmp/log/in.telnetd.

You can stop the trace using the "-d" option with traceit.

Example:

$ /etc/traceit –d in.telnetd

Invocation of UWIN commands from other applications.

UWIN commands can be directly invoked from any other applications.

Setting Path variable

The PATH should include the UWIN install directory. Ie. $UWIN_ROOT/bin. Thereby all the UWIN commands like ls,ps etc can directly be invoked from other applications.

Invoking shell scripts:

Shell scripts can be invoked by passing it as an argument to the ksh.exe from $UWIN_ROOT.

Typical call would be

$ /c/uwin/bin/ksh.exe install.sh

Posix thread support in UWIN

UWIN supports POSIX threads (pthreads). Any application which invokes pthreads and have a multithreaded application can be written and compiled on UWIN. The maximum number of threads on any particular system is limited by the resources available in the given system.

Archiving using “Pax”

Pax command in UWIN can be used to read, write or list the file archives. Pax can handle several formats like ansi, asc, binary, cpio, cab ibmar, mime, pax, etc. You can get the list of the supported formats from the man page of pax.

This section familiarizes you with some of the commonly used form of pax command.

List the file archive

"-v" option to "view" the contents of the file archive.

Eg.

```
$ pax -vf  sources.pax.gz
```

Extracting a pax file

"-r" option of pax can be used to extract the files from a file archive.

E.g.

```
$ pax -rvf sources.pax.gz
```

Create file archive

"-w" option can be used to create a pax file. You can pax a file, directory, or list of files (using regular expressions).

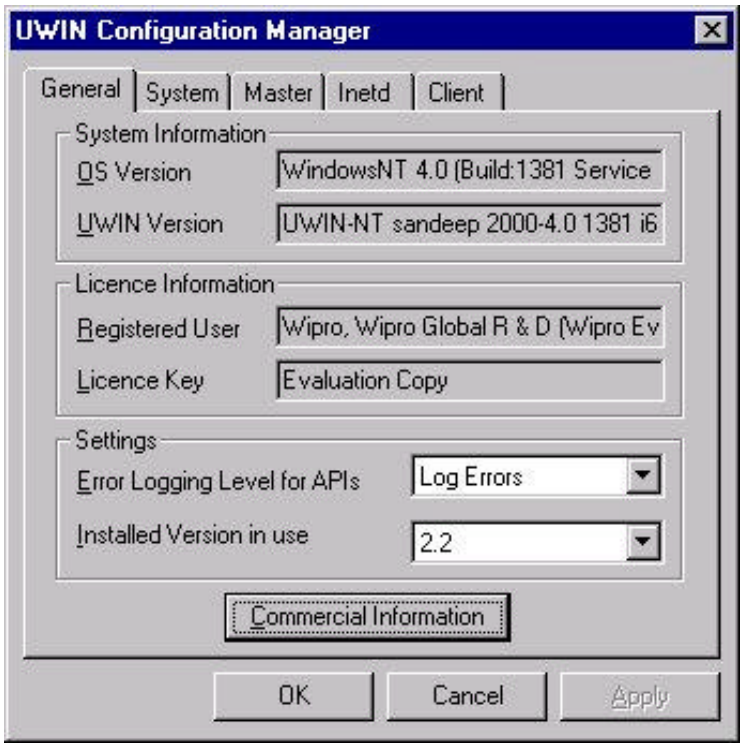
Command Syntax :

```
pax -wvf <pax-file-name> <files-to-be-archived>
$ pax -wvf sources.pax.gz /tmp/*log
```

The pax command retains the path information and the permissions when the archive is created and the same is restored when the file archive is extracted.

UWIN Control Panel Applet:

UWIN control panel applet is an icon in the control panel used to configure & fine-tune UWIN.



The UWIN control panel applet is also know as UWIN configuration manager as this gives the option of fine tuning the UWIN resources. There are around five tabs in this applet each one is having its own unique features.

General:

The general tab is the first tab we encounter when we go to the control panel applet. This tab gives the system description.

It gives the OS on which UWIN is mounted, along with the version, build number, service pack information of the OS etc.

Windows NT 4.0 (Build:1381 Service Pack 6) / NTFS (case preserving) / Intel i686(1 processor)(64884 KB)

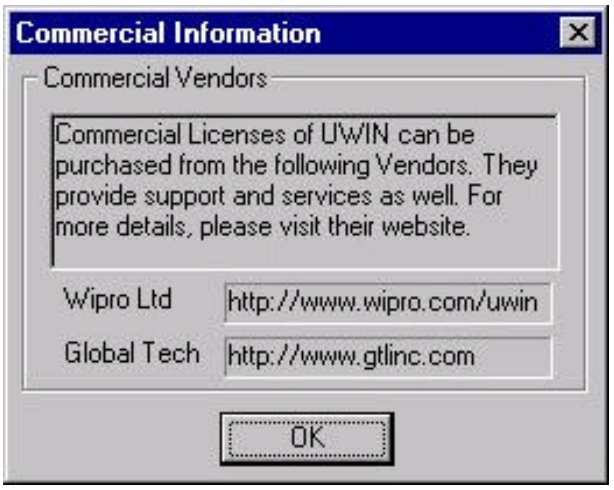
A snapshot of a typical entry is as shown above.

This also gives the version & other details of UWIN release, which is mounted in the particular system. This shows all the details which one get after executing uname –a on UWIN.

The license information gives the details specified during installation of UWIN. This also says about the license status, whether it is an evaluation copy or a commercial copy.

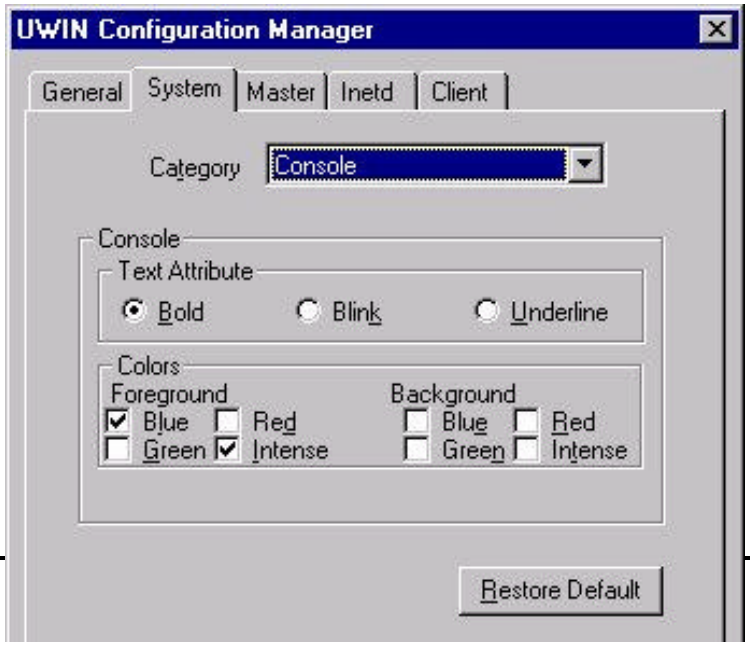
The settings option is for switching between different versions of UWIN installed. This changes the working directory for each version of UWIN.

The commercial information button pops up one more dialog,



This gives the information for commercial purchase of the UWIN product.

System:



In the system tab there are six categories, which can be configured

Console:

This option is used to change the setting of the interface console. The colors set for the fonts can be changed according to the users liking. All the changes associated with the console can be configured using this option. The restore default button is used to get back the default settings used by the system.

Message queues:

This option is used to set the limits for the message queue.

This sets the upper limit on the number of identifiers, maximum queue size & the maximum size of a message in a message queue.

Resources:

This options sets the limits on size of FIFO's, maximum number of files a process can open, maximum number of processes that can be created & the maximum no. of Sid's. All the changes made through the applet will be reflected once UWIN is rebooted.

Semaphores:

This option deals with setting up the upper limit on the number of semaphores in the system, number of operation for each semaphore, maximum number of semaphores per identifier etc from this option of the configuration manager.

Shared memory:

In this option the maximum number of identifiers for a system, maximum segments per process; maximum size of the shared memory & also the minimum size can be set.

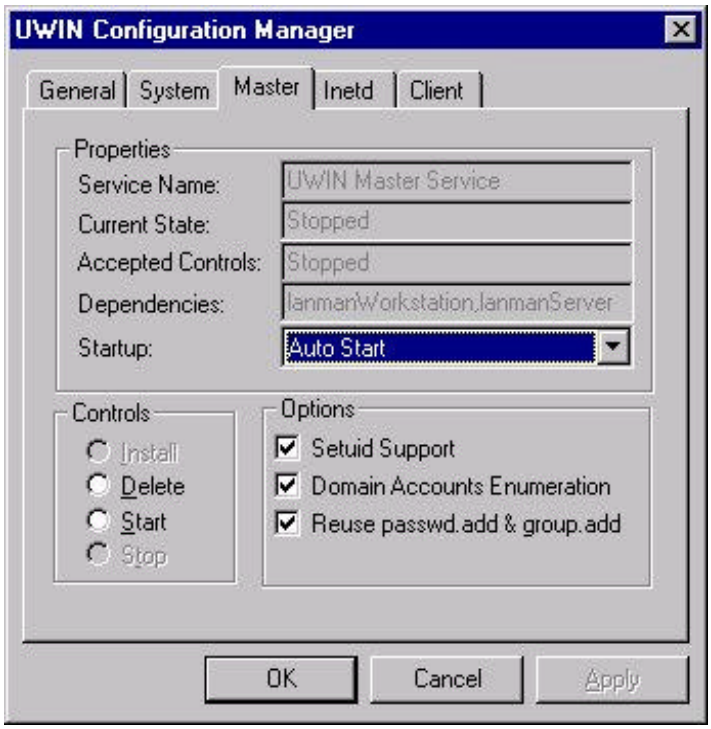
Miscellaneous:

This gives the option of making UWIN case sensitive. By clicking this option & rebooting UWIN, the file system will be case sensitive.

Master:

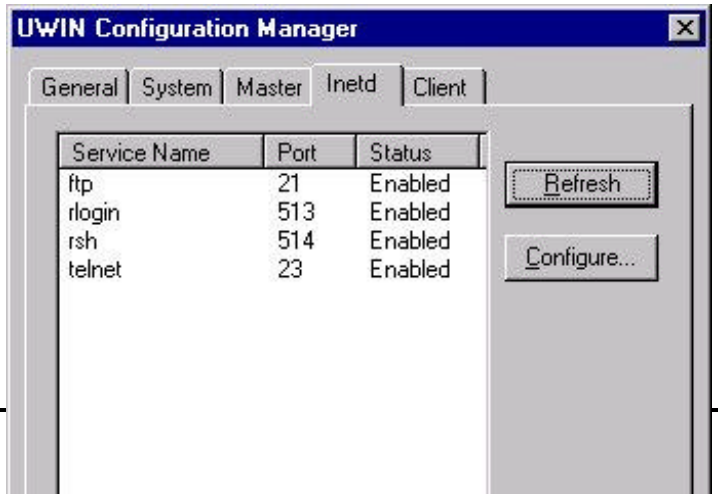
This tab deals with the UWIN Master Service (UMS) of UWIN. The details for this tab are obtained during the installation of services for UWIN. This specifies the current status of the Ums service, the mode of initiating the service; it can be automatically started whenever UWIN starts or manually started from the console or disabled completely. The controls provided make it to stop or delete a started service. They also install the service incase if it is not enabled automatically.

In the options tab, the domain accounts enumeration button, if checked, will take the details of all the accounts in that domain & create home directories for each of the account.



inetd:

This tab controls the Internet super server of UWIN. This lists the telnet, rsh, rlogin & ftp along with their status & the number of the port through which these services are enabled. This forms the most commonly used inet daemons in UWIN & this options list only these four daemons. This gets the details from inetd.conf files and lists them here & no other service can be added on to this list.



Whenever inetd starts, it searches for an entry in inetd.conf, this is found in /etc, if a service is not listed in this then that service will not be invoked at all. All the changes in the inetd.conf are updated through a script file inetdconfig.sh which is found in the /etc directory.

Configure - This option is used to configure the services. Through this we can change the port on which this service is running, we can also enable or disable the service using this dialog.

Client:

This option is used to control the UWIN Client service (UCS) of UWIN.

Installed Clients:

This window displays all the clients for which UCS has been installed.

The list will be generated dynamically as soon as a UCS service is installed for a particular user.

Properties:

This window displays the properties of each of the user for which the UCS is installed. This displays the property of a particular UCS user, which is being selected from the Installed client's window.

Control:

The option for deleting the UCS for any of the clients listed in the installed client window is provided here. Select a particular user from the window & click on the delete button and say ok, the service will be deleted.

Installation wizard:

Using this wizard the user can install the Ucs service for the accounts listed.

Installing UCS:

The following procedure can be followed for installing the UCS on a system running Uwin with the installation wizard of the Uwin control panel applet.

The installation wizard takes you to UCS installation menu, which gives a list of all the accounts on the local machine. The listing is in the form of a tree structure. For those accounts for which the ucs service is installed, it will not have an entry in the tree & only those for which it is yet to be installed, the list will be displayed.

For installing the service the user must click on any account for which the service needs to be installed & click on the install button. The user must provide the correct password for the particular account in order to install the UCS service. A confirmatory message will be popped up for each successful creation of UCS service. The Installed client will be displayed in the list of installed Clients menu in the client tab of the control panel.

Conversion Utilities:

html2rtf:

Description

HTML2RTF is a utility, which converts from HTML to RTF (Rich Text Format). RTF is a document format, which most word processing programs can read.

Purpose

HTML2RTF is useful for when you want to make high-quality hardcopies of HTML documents.

For example, if you want to hardcopy a Web page, but want to double-space it, you'll find that your browsers probably don't support this. But if you use HTML2RTF to render the Web page as RTF and then import this RTF file into your word processor, you can then adjust the line spacing (or page margins, etc.) before you print.

HTML2RTF is also useful when you want to edit an HTML file together with word processor documents -- e.g., when you want to paste the content of a Web page into a word-processor document. An RTF rendering of the HTML file should be editable like any other word-processor text.

Usage

Invoke this utility, specifying on the command line the files to render. For example, use:

```
html2rtf foo.html
```

This will produce an output file called foo.rtf. Multiple filenames can be specified, and paths as well.

```
html2rtf ../bar/baz/foo.html quux.html
```

This renders ../bar/baz/foo.html as ../bar/baz/foo.rtf, and quux.html as quux.rtf

The filename's "html", "HTML", "htm", or "HTM" extension is stripped off, and "rtf" is added on. If the filename has no "html", "HTML", "htm", or "HTM" extension, "rtf" is added to the end of the full filename.

html2db:

Description

html2db extracts a flat file database from tables in the input html *file*s. If *file* is not specified then the standard input is read.

Usage

Invoke this utility, specifying on the command line the files to render. For example, use:

```
html2db foo.html
```

troff2html:

Description

troff2html converts files written in the *troff* input language and converts them to Hypertext Markup Language (HTML). Its main use is for man2html that converts *man* pages into HTML.

Running troff2html

The basic syntax is simply troff2html inputfile.me. Make sure you're in a separate, writable directory; troff2html assumes freedom to write a lot of files. troff2html will process all files on the command line as a single document.

A simple document will probably be best translated as:

```
troff2html -nosplit -notoc simple.me
```

which gives a single page document simple.html with no table of contents. It's a good idea to tell troff2html explictly -nosplit -notoc if that's what you want; strange things can happen otherwise.

A bare troff2html will give the following usage statment:

usage: troff2html [arguments] input files ...

Refer to the man page of these commands to get information on different options supported.

Chapter 4

Services and Daemons

This article is meant for UWIN running on Windows NT environment only.

Services are specific to the Windows NT environment. They are processes that can be automatically started by the OS at system startup. There is an option to run a Service under a particular user account, or the system account, also know as LocalSystem. Services can be started, paused, stopped through the Service Applet in the Control Panel.

There are two services specific to UWIN. They are:
UWIN Master Service (UMS)
UWIN Client Service (UCS)

UWIN Master Service (UMS)

The UWIN Master Service starts up with the system boot, and keeps running in the background. Running the following command stops the service

```
net stop uwin_ms
```

from the command prompt . It can be restarted with the command

```
net start uwin_ms
```

It provides the following services to UWIN and its applications.

Creation of /etc/passwd and /etc/group files

The UMS is responsible for creation of /etc/passwd and /etc/group files. Every time the NT system boots up, UMS queries the user database and recreates these files with the NT User database information.

The user information to be stored in these files is obtained from the NT User Database. If the NT system is part of a WorkGroup, only the Local user database is queried, and the passwd and group files contain the local accounts only. If the NT system is part of an NT Domain, then the Domain database is also queried along with the Local database and the passwd and the group file contains both Local and Domain accounts. For a Domain Controller, only one set of Domain accounts is present in the passwd and group files. This Domain accounts enumeration feature can be controlled from the UWIN Control Panel applet.

Any changes made to the /etc/passwd or /etc/group files will get over-written at the next system reboot. Hence, for creating user accounts, of modifying user information, the NT User Manager (musrmgr.exe) should be used. Any changes made to the user database through the NT User Manager get reflected in the passwd and group files only after the next reboot. To reflect the changes in the User database in the passwd and group files, /etc/mkpasswd can be used.

Starting of daemons

When UMS starts up, it runs the script /etc/rc. This script is used to execute tasks that are scheduled to run at system startup. inetd daemon is started through this script. The other tasks done in the rc script are:

- clear out utmp and utmpx files
- truncate the /tmp/uwin_log file
- mount any filesystem specified in /etc/fstab
- run inetd
- run sshd (if available)
- run crontab

Setuid functionality

A user belonging to the Administrators group can make a setuid() call. UMS provides the functionality through which setuid() calls can be made by other applications.

UWIN Client Service (UCS)

A UWIN Client Services (UCS) has to be installed for every user to whom a setuid() call is to be made. Telnet-ing to a user account in UWIN, automatically installs a UCS service for that user.

UCS can be installed by running the following command:

```
$ /etc/ucs install <username> <passwd>
```

where <username> is the account for whom a UCS service is to be installed, and <passwd> is the password of that user.

For uninstalling the UCS, execute the following command:

```
$ /etc/ucs delete <username>
```

where <username> stands for the account for which ucs service is to be deleted.

If you change the password for a user account for which a UCS services is already installed, you need to delete and reinstall the UCS service with the new password.

Working with Remote tools

UWIN includes telnet, ftp, rlogin and rsh daemons. These daemons are invoked by inetd, the Internet super-server. UMS runs /etc/rc in startup, and this script starts up the inetd daemon, which in turn invokes the other daemons.

Connection-oriented services are invoked by the inetd each time a connection is made, by creating a process. Datagram oriented services are invoked when a datagram arrives; a process is created and passed a pending message on file descriptor 0.

inetd uses a configuration file, /etc/inetd.conf which is read at startup and, possibly, at some later time in response to a HUP signal. The configuration file is ``free format'' with fields given in the order shown below. Continuation lines for an entry must being with a space or tab. All fields must be present in each entry.

Service name	must be in /etc/services or must name a tcpmux service
Socket type	stream/dgram/raw/rdm/seqpacket
Protocol	must be in /etc/protocols
wait/nowait	single-threaded/multi-threaded
User	user to run daemon as
Server program	full path name
Server program arguments	maximum of MAXARGS (20)

TCP services without official port numbers are handled with the RFC1078-based tcpmux internal service. Tcpmux listens on port 1 for requests. When a connection is made from a foreign host, the service requested is passed to tcpmux, which looks it up in the servtab list and returns the proper entry for the service. Tcpmux returns a negative reply if the service doesn't exist, otherwise the invoked server is expected to return the positive reply if the service type in inetd.conf file has the prefix "tcpmux/". If the service type has the prefix "tcpmux/+", tcpmux will return the positive reply for the process; this is for compatibility with older server code, and also allows you to invoke programs that use stdin/stdout without putting any special server code in them. Services that use tcpmux are "nowait" because they do not have a well-known port and hence

cannot listen for new requests. Comment lines are indicated by a `#' in column 1.

Any user-defined service can be started by the inetd by making an entry in the configuration file. inetd has to be running all the time for the services to come up.

Using Telnet andin.telnetd

The /etc/in.telnetd daemon allows users to do a logon to the NT machine from the network. When a user logs into a NT/UWIN system using telnet, the default shell is as specified in /etc/passwd file. The UWIN's in.telnetd requires users to login using their Windows NT domain and their login name. Login names are case-sensitive, and must be entered as they were entered into the User Manager.

If the NT system is part of a NT Domain, a telnet user can either log into the LocalSystem or the Domain. For logging into the LocalSystem, only the username (in the Local System) should be used. For logging in the Domain, the Username should be given as "Domain/Username" (where username is the name of the user account on the Domain).

ftp and in.ftpd

This service allows users to transfer data over the network. The corresponding daemon is /etc/in.ftpd. For this daemon to function, a UCS service has to be installed for the user in whose account, a ftp login is made.

rlogin and in.rlogind

This service allows users to login from a remote machine. The corresponding daemon is /etc/in .rlogind.

rsh and in.rshd

This allows users to execute a command from a remote machine. The daemon is /etc/in.rshd.

rcp

The "rcp" command copies files between machines. Each file or directory argument is either a remote file name of the form `rname@rhost:path'', or a local file name. Rcp uses the in.rshd.

Configuration for rlogin / rsh / rcp

This section gives you a few checks and guidelines with examples for configuring and working with rlogin/rsh/rcp.

1. ums & inetd should be running on the UWIN system.

You can start the ums using the command "net start uwin_ms" from MS-DOS or UWIN prompt.

2. ucs should be installed for the user account to which rlogin/rcp/rsh command is to be executed. "ucs" can be installed using the UWIN control panel or by telnet-ing to the user account.

3. Have appropriate entries in the ".rhosts" file in the user directory. The ".rhosts" file should be put in the "HOME Directory" of the user to which you want to rlogin/rsh/rcp. The file should have the entries in the following format

<Machine-name/IP address> <user-name>

For example,
if you are trying to rsh to an user account "user1", on the NT machine from an UNIX machine "unix-1" and user account "unix-user1" then you will have to have the following entry in the ".rhosts" file in Homedirectory of "user1" ie. "/home/user1".

unix-1 unix-user1

If you are trying to rcp/rsh/rlogin from UWIN to UNIX.

Make sure you give even the domain account name with your user id in the .rhosts of the UNIX machine.

just type "logname" on the ksh prompt and put the same string as your user id in the .rhosts file on the unix machine

Typical rlogin commands

$ rlogin –l<user-id> <machine-name/IP address>

E.g.:
rlogin to local user account

$ rlogin –ljack godzilla

rlogin to a domain account.
Domain :NT_POSIX.DOM & User-ID : bob

$ rlogin –lNT_POSIX.DOM/bob godzilla

With proper entries in the .rhosts file and the above mentioned configuration setup, you should be able to login without being prompted to enter the password.

Typical rsh commands

$ rsh –l<user-id> <machine-name/IP address> <command>

E.g.:
rsh to local user account

$ rsh –ljack godzilla pwd
/home/jack

rsh to a domain account.
Domain :NT_POSIX.DOM & User-ID : bob

$ rsh –lNT_POSIX.DOM/bob godzilla pwd
/home/NT_POSIX.DOM/bob

Typical rcp commands

$ rcp <user-id>@<machine-name/IP address>:<path of the file to be copied> <target-path>

E.g.:

copying a file “samp.c” from jack’s home directory to the current directory

rcp to local user account

```
$ rcp jack@godzilla:samp.c .
$
$ls
samp.c
```

copying a file “samp.c” from bob’s (domain account) home directory to the current directory

Domain :NT_POSIX.DOM & User-ID : bob

```
$ rcp NT_POSIX.DOM/bob@godzilla:samp.c .
$
$ ls
samp.c
```

CHAPTER 5
Terminal Emulation

The terminal is the console window displayed when you run an UWIN shell. This chapter describes the UWIN terminal and the ways in which the appearance of a terminal can be customized.

The UWIN console terminal emulator supports vt100 terminal capabilities. It also supports color using ANSI Escape Sequence. The environment variable, TERM decides the terminal emulation. The environment variable TERMINFO points to the terminfo database. The following commands in /etc/profile initializes the terminal.

```
$ export TERM=vt100
$ export TERMINFO=/usr/lib/terminfo
```

Customizing the Color and Size of the Terminal

It is possible to resize the terminal, change the colors, enable/disable scrolling. In Windows NT, under UWIN environment, it can be done by adjusting the properties in the Console Property sheet.

Steps to customize the terminal:

1. Open a UWIN shell
2. Click the icon on the top left corner
3. Select the Properties field
4. Select the appropriate tab to adjust the settings
   - To change the size of the window, select the Layout tab and enter the required sizes for the buffer and the window
   - To change the colors of the text and the background, select the color tab and choose the required colors from the palette
5. Click on the OK button
6. Windows NT displays a dialog box asking whether you want the changes to affect the current window only or save properties for future windows with the same title.
7. Choose as per requirement
8. Click OK
9. Left click on the UWIN console to inform UWIN of the new changes

An alternate method of changing the size of a window is using the command stty that is described later. In Windows 9x, stty is the only way to adjust the size of a window.

Points to be noted:

- The user has no control over the color of the cursor.
- Changing the values of Popup Text and Popup Background will not affect in any manner.
- In Windows 9x, it is advised to set the font size to some value instead of " auto".

Enabling Scrolling

It is possible to enable scroll bars on a UWIN terminal. The only setting that needs to be done is to make the buffer size more than that of the window size. This is true for Windows NT only.

Cut and Paste

Steps to cut and paste a block of data:

- Drag the left mouse button to select the text in the shell window. The selected text appears in reverse mode. Once the left button is released, the text is copied to the clipboard.
- Click the right-mouse-button (for a 2-button mouse) or the middle-button (for a 3-button mouse) to paste the selected text from the clipboard to the terminal.

Title

You can specify the text displayed in the title bar of an UWIN terminal window. This is often used to display the currently executing command in the title bar. The title string can be up to 512 characters long.

There are two ways of setting the title

- With the title command.

Ex 1: To set the title of the screen to "my_screen", execute the following command

```
$ title my_screen
```

Ex 2: To set the title to the present working directory, execute the following command

```
$ title `pwd`
```

- Programmatically by sending the following Escape Sequence to the console.

  Esc]0;<title string> ^G

Where <title string> contains the string.

VT100 Attributes

The attributes associated with the characters are bold, blink and underline. These are not implemented in the literal sense but are implemented in the form of different colors for each attribute. Some default colors have been assigned to these attributes but can be configured as per personal preference, by the user through the registry editor.

The keys that contain the values of the attributes are:

- BlinkText
- BoldText
- UnderlineText

The path for these keys is:
HKEY_LOCAL_MACHINE/Software/AT&T Labs/UWIN/<*CurrentVersion*>/Console
where <CurrentVersion> stands for the current version of UWIN.

The pre-assigned values of the attributes are:

Key Name	Color	Value
UnderlineText	Green	10
BoldText	Blue	9
BlinkText	Red	12

If there is another key, NewVT under Console, make sure that the value of this key is always 1.

Steps to be followed to adjust the attribute values, for example, to change the value of bold attribute:

1. Run Regedit.exe from the command prompt.
2. Click on the Console key following the path described above.
3. Double-click the sub-key BoldText and enter the new value.

New value = Foreground color + Background color

The values of the foreground color and background color can be obtained from the table shown below. If bold text is to be displayed as blue (1) in the foreground and light green (32) in the background, then the value of BoldText is 1 + 32 = 33.

Color	Foreground (decimal)	Background (decimal)
Black	0	0
Blue	1	16
Light Green	2	32
Turquoise	3	48
Red	4	64
Pink	5	80
Yellow	6	96
Grey 25%	7	112
Grey 50%	8	128
Dark Blue	9	144
Green	10	160
Teal	11	176
Dark Red	12	192
Purple	13	208
Dark yellow	14	224
White	15	240

stty command

stty command displays all the terminal settings .

Ex:

```
$ stty
speed 19200 baud, 40 rows, 79 columns
kill = ^U;
swtch = ^@; susp = ^Z;
-parenb -parodd -cread -hupcl -cstopb –clocal -ignbrk -brkint -
ignpar -parmrk -inpck -istrip -inlcr -igncr -iuclc -ixany -ixoff -
imaxbel -iexten-echonl -xcase -noflsh –tostop -olcuc -onlret -ocrnl -
onocr -ofill -ofdel
```

stty –a gives a more detailed list . This command is also used to adjust the terminal settings.

The terminal window size can be changed with the rows and columns arguments.

Ex: To set the size of the window to 50 x 80, run either of the following commands.

```
$ stty cols 80 rows 50
or
$ stty columns 80 rows 50
```

The same effect can also be achieved with the following two commands.

```
$ stty cols 80
$ stty rows 50
```

It should be noted that the number of columns displayed inside the property sheet is one more than the UWIN column size which is evident by executing the command

```
$ stty –a
```

Apart from setting the size of the window, this stty command can also be used to set other parameters like susp, kill, stop, etc.

Ex: To kill a line, the control character is ^U. It can be changed to ^B by executing the following command:

```
$ stty kill ^B
```

A more detailed explanation of each argument can be obtained from the man page of stty.

It should be noted that sometimes the parameters of the terminal interface may get disturbed because of which the user may get irritating behavior from the terminal like -

- Characters typed may not be echoed.
- Cursor may not move to a new line on pressing carriage return from ksh, etc,

In order to rectify this problems you need to execute the following command.

```
$ stty sane ^J
```

Chapter 6
Utilities and APIs

Utilities

There are more than 300 utilities supported by UWIN. Some of them are listed in the Appendix A.

Some of the shell functions provided with UWIN are as follows. They are located in /usr/fun directory.

keybind, vi_keybind, emacs_keybind

Tab completion of filenames in Ksh is possible using keybinds. Command history can be searched through the use of arrow keys. If you are using Ksh in vi edit mode, use the vi_keybind feature. Else if you are using Ksh in emacs mode, use emacs_keybind feature. By default, UWIN starts the ksh shell in vi edit mode, and /etc/profile automatically calls vi_keybind.

You need to call either of the function from your shell to enable the keybind features

For vi editing features

```
$ vi_keybind
```

and for emacs editing feature

```
$ emacs_keybind
```

Now, you will be able to use arrow keys for searching through the command history, and Tab key for filename completion. The keybind related files are stored in /usr/fun.

popd, pushd & dirs

popd and pushd enables you to toggle between directories. pushd pushes the current directory on to the stack and popd is retrieve is back from the stack. The top of the stack is the present working directory. dirs displays the content of this stack.

Ex:

```
$ pwd
/usr/src/lib/libposix
$ pushd /d/winnt/system32/
/d/winnt/system32  /usr/src/lib/libposix
$ pwd
/d/winnt/system32
```

```
$ pushd
/usr/src/lib/libposix /d/winnt/system32
$ pwd
/usr/src/lib/libposix
$ pushd
/d/winnt/system32 /usr/src/lib/libposix
$ pwd
/d/winnt/system32
$ pushd
/usr/src/lib/libposix /d/winnt/system32
$ pwd
/usr/src/lib/libposix
$ dirs
/usr/src/lib/libposix /d/winnt/system32
$ popd
/d/winnt/system32
$ pwd
/d/winnt/system32
$ dirs
/d/winnt/system32
```

ie

An inline edit command named ie has been added to provide ksh style editing and history to any interactive line oriented command. For example, ie cat will allow command line editing of the input lines to cat.

title

The title utility allows you to change the title in the top of the window.

Ex:

```
$ title “UWIN Shell prompt”
```

Changes the title of the window to “UWIN Shell prompt”. You can also put the result of a command as the title.

Ex:

```
$ title `pwd`
```

This command results in changing the title to the present working directory.

Configuration Management utility

The version maintenance can be done using the RCS source control commands like ci and co. The functionality of these commands is similar to that on Unix. It is possible to get any intermediate version

or a file at any time, once the version is checked in. UWIN include rcs for revision control.

APIs

All of the APIs in UNIX are implemented here. The list of APIs implemented is found in Appendix B.

Chapter 7
Differences between Unix and UWIN

There are some differences between the UWIN environment and a traditional UNIX operating system. The major differences are with reference to:

- Pathnames that can contain drive letters
- Administrator group
- Access of Registry through the file system

Pathnames on UWIN

Note the following while referring a file in UWIN

- A pathname that begins with / is taken to begin at the root.
- A pathname beginning with more than two slashes are treated as if it begins with one slash.
- Allows Universal Naming Convention (UNC) for files.
- Allows drive names in the path names.

The Administrator Group and Appropriate Privileges

Specific types of privilege are essential for actions such as file system access, certain process control actions (sending signals to other processes), changing the effective user ID or group ID of a process, etc. Keeping this in view, the users of UWIN can be broadly divided into the following two categories:

- Administrators group
- A user

The administrators group may contain a single person or a group of persons. The administrators group is a group of users with special privileges who are allowed to change the file's permissions including, but not limited to successfully invoking the setuid(), setgid(), sending signals to other processes, etc. A user who does

not belong to the Administrators group cannot perform these operations.

Depending on the resource access privileges, the UWIN users can also be classified as follows:

- Domain user
- Local user

Accessing Registry through UWIN File System

This is a UWIN specific attribute that can not be observed in the case of Unix. Windows registry is treated as a file system with registry keys treated as files and keys that have sub-keys treated as directories. The tools supplied with the toolkit can be directly used on the registry keys. The registry is automatically mounted under /reg during startup.

Finding UWIN Version information

UWIN version information is given by the version of the “posix.dll” in the /sys directory. This information can be obtained using the following command from ksh prompt.

$ what /sys/posix.dll

Typical output of the command looks like the one given below.

$ what /sys/posix.dll
Vmalloc (AT&T Labs - kpv) 1999-07-01
UWIN Version 2.9.1(03082001)
Wipro UWIN Commercial License

UWIN version information can also be obtained from the “General” tab of the UWIN control panel applet.

There are two ways to find the versions of UWIN running.

[This also gives whether the package installed is an Evaluation or Commercial version.

Chapter 8
Troubleshooting

Q: Tab completion and arrow-keys not working.

A: Execute vi_keybind at the shell prompt or include the same in /etc/profile.

Q: Problems with telnet, ftp, rlogin or rsh

A: When you are not able to telnet, ftp, rlogin or rsh and an error Unable to connect to remote host: Connection refused, comes up, make sure that inetd.exe is running on the host machine. There are two ways of starting inetd :

```
$ /etc/inetd &
```

Start UMS with the command

```
net start uwin_ms
```

It automatically starts up inetd.

The log messages during installation are recorded in the file /tmp/install_log. The log messages for the UMS and UCS services are logged in the files /tmp/ums.out and /tmp/ucs.out. When UWIN is being used, all failures, exceptions and access violations are logged in /tmp/uwin_log.

UWIN and Oracle

This section gives you tips on tackling some common problems when UWIN and Oracle co-exist on the same system.

Init.exe start up problem

Problem description : “init.exe” does not start up on reboot when UWIN and Oracle co-exist on the same system.

Workaround:

While running UWIN on a machine that has Oracle installed, the UWIN master service needs to be started before any Oracle services start. Stop all the Oracle services (from the services applet in control panel) and start the UWIN master service by doing the following:

```
$ net start uwin_ms
```

After this, start the Oracle services (from the service applet in control panel)

Setting ORACLE variables

The ORACLE_HOME has to be set using MS-DOS notation versus UWIN/UNIX notation for SQL*Plus and the Pro*C pre-compiler .

i.e. for example
ORACLE_HOME="C:\ORAWIN817"
versus
ORACLE_HOME="/C/ORAWIN817".

An alternative is to put the name of the variable into MS DOSPATHVARS. In this way the value will be in UNIX format for UNIX processes and in MS-DOS format for native processes.

```
$ export ORACLE_HOME=/C/ORAWIN817
$ export DOSPATHVARS+=' ORACLE_HOME'
```