Wipro UWIN

Porting Guide

Wipro Ltd
1995, El Camino Real,
Santa Clara,
CA – 95050, USA
uwin@wipro.com
http://www.wipro.com/uwin

CONTENTS

Chapter 1
Getting Started

Introduction

This section deals with the minimum modules of UWIN required to port an application from Unix to Windows NT/9x. Also, the different types of services and support offered by Wipro to the developer.

Requirements to port an application

The minimum requirement to port an application from Unix to Windows system using UWIN is to get the UWIN base toolkit and UWIN-SDK (UWIN Software Develo pment Kit) installed. In addition to this, the developer should have either GNU gcc or Microsoft Visual C++ compiler to compile the application.

For the list of prices regarding each of the UWIN packages either visit the website http://www.wipro.com/uwin or send a mail to uwin@wipro.com

Services and Support from Wipro

Wipro has dedicated software professionals who can provide services and support to help the developer in porting the applications. Below are the details for contacting the team.

Services

Wipro provides the following services.

- UWIN – Getting started
- Migration support services
- Turnkey Migration services
- Training program

UWIN - Getting Started

This is a short term consulting service aimed at customers who have in-house Unix expertise and want to apply the same for porting the application from Unix to Windows systems using UWIN.

This program will impart the developer a thorough knowledge of the following issues.

- Issues in porting Unix applications to Windows which include the dependencies, hurdles, constraints, performance
- Cost benefit analysis

Migration Support Services

This program will assist the developer in porting the Unix applications to Windows NT/9x. This program will look into the problem at hand that has already been carried out by the migration team of the customer, and advise them in formulating the future course of action.

Turnkey Migration Services

This program is focused towards organizations that would hire a Migration Solution Provider, like Wipro for the entire migration activity on a turnkey basis.

This comprehensive program covers the following issues:

- Drawing up the Requirement Specification for the Migration task
- Identifying key phases involved in the project
- Preparing a Detailed Plan for carrying out the Migration

Training Programs

The training programs offered by Wipro are targeted at providing a good understanding of the product and its features.

The following types of training programs are available:

- UWIN from a Users perspective
- Migrating applications to Windows using UWIN
- Interoperability between Unix and Windows

UWIN from a Users perspective

This training program is targeted towards users who would like to use UWIN as a 'Unix utilities for Windows' Toolkit. UWIN will be presented to the users as a tool with a rich set of Unix commands, shells and utilities on Windows. Stress will be given on the following topics:

- Usage of shell scripting features of UWIN using Kornshell93 (ksh93)
- Scheduling and automating tasks using cron jobs
- Interfacing with the underlying Windows OS features using UWIN
- Using the UWIN Terminal Emulator
- Working with UWIN X11 applications (xterm, etc)

- Setting up sendmail mail server and bind DNS server on UWIN
- Improving cost-effectiveness and productivity using UWIN

Migrating applications to Windows using UWIN

This training program is targeted to those who would like to use UWIN as a Unix application development platform on Windows. This program will cover the basic understanding of UWIN from users' perspective apart from the application development aspects.

The topics covered are:

- UWIN as a Unix application development platform on Windows
- Using the UWIN cc compiler with a native Windows compiler
- Working with UWIN header files and libraries
- Developing X11 applications on Windows
- Creating shared libraries and Windows DLLs using UWIN
- Using UWIN GNU gcc compiler on Windows
- Deploying UWIN applications on Windows based systems

Interoperability between Unix and Windows

This training program helps users in exploring the issues involved in working with both Unix and Windows operating systems. The different issues ranging from user-interaction to application development will be covered. All available Unix-NT Integration and migration products will be discussed.

Chapter 2
Application Porting

This chapter describes the different steps involved in porting an application from Unix to Windows NT/9x using UWIN.

Overview of the Porting Process

Before porting an application, the developer has to prepare the application for porting. The process of porting involves the following steps:

- Preparing the application for porting
- Preparing the application for building
- Dealing with missing header files and libraries
- Compiling and linking the Application
- Debugging the application
- Packaging and deploying the application

The responsibility of the developer in each step is described in detail in the subsequent paragraphs.

Preparing the application for porting

Before porting an application, the developer has to decide the following:

- Determine the possibility of porting
- Access the source files
- Select a work environment

Determine the possibility of porting

Some applications may require some other tools to run them, which might not be available on UWIN. Under such circumstances, before porting that application to Windows NT/9x, the developer should make sure that the required tools are already ported on UWIN. Otherwise, it will not be possible to execute the ported application on UWIN.

Already a few tools have been ported to UWIN, the list of which can be obtained from http://www.wipro.com/uwin/. If required, the developer can download them free of cost.

Access the source files

One of the commonly faced problems while copying source files to target operating system is line termination. The line termination in case of Unix is a line-feed character whereas in Windows, it is a carriage return and a line-feed. In order to convert the file to Unix format, the developer can make use of the utility /bin/nocrnl.

Example:

If my_app.c file is copied to WinNT/9x, the execution of the following command will convert the file my_app.c to Unix format.

```
$ nocrnl my_app.c
```

Transferring the source files to the target operating system can be achieved through ftp or rcp and the archive can be expanded using pax or tar. The information for these commands can be obtained from the appropriate man pages.

Select a work environment

The developer has to decide the environment to which the application is to be ported. It can be Windows NT workstation or server, Windows 95, or Windows 98. UWIN ensures that when an application is ported on one work environment, it runs on all other environments as long as the underlying platform supports the features that the application requires. Some features like security is not available on all work environments. The developer can opt for gcc or cc (that internally calls Windows native C compiler) to compile the application.

Preparing to Build

If the application involves a makefile, it has to be converted to Unix format so that UWIN make/nmake recognizes it properly. The format conversion can be achieved by the utility /bin/nocrnl as described in the section *Access the source files* above. UWIN comes with AT&T nmake and GNU make. Hence the developer may need to make changes to the makefile so that is can be interpreted properly.

For application that does not have makefile, it is not necessary to convert all source and header files to Unix format.

Dealing with header files and libraries

While porting the application, the developer may have to resolve issues that arise due to the UWIN provided and the own headers and libraries of the application. During the time of compilation if any conflict arises, the compiler search path should be given so that it first takes the local headers/libraries and then the UWIN headers/libraries or vice-versa.

Example:

```
        $ cc –I<path> -o test.exe test.c
```

where <path> is the place where the header/libraries are present.

Compiling and Linking Applications

While compiling an application on UWIN, developer might need to make minor changes in the source code in some cases. UWIN aims at 'single source, multiple deployment' concept. However, at times the code being compiled/linked may not be X/Open compliant. In such cases, the developer must do some modifications in the code to make it X/Open compliant. The amount of effort that the developer needs to put to make this change depends on what percentage of the code is not compliant.

UWIN makes it possible to mix Win32 and Unix APIs at the source level. However, the developer is advised to make use of only Unix APIs to get the desired result. If the APIs are mixed, a single source base for both Unix and Windows environment cannot be maintained.

Debugging applications

If it is required to debug an application, the binaries should be built with the debug option.

Example:

If my_app.c is the application that needs to be debugged, then build the binary as follows:

```
        cc –g –o my_app.exe my_app.c
```

Now, my_app.exe contains the debug information. If the application has been compiled through a makefile, then putting the below line in the makefile will generate a debug version of it.

```
        CCFLAGS=-g
```

After this, the application can be debugged using Microsoft Visual Studio IDE. For more information on debugging an UWIN application under the IDE, please refer to Chapter 4: Debugging Using Visual C++ IDE in the developer guide.

The application can also be debugged using GNU gdb if the binary is built using GNU gcc compiler.

The UWIN trace command can be used to debug an application at the system call level.

Example:

To trace the Unix APIs that my_app.exe calls, the below command can be used.

```
$ trace –it –o outfile.log my_app.exe <arguments>
```

where <arguments> are the command line arguments that the application takes when run normally.

The output of trace command will now be available in the file outfile.log. The file outfile.log will contain all the Unix API calls made by my_app.exe along with the values of arguments passed and the return value from them.

To trace a daemon (like inetd.exe) and applications started by a daemon the below method must be used.

To start the tracing use the command:

```
$ /etc/traceit <daemon_name>
```

where <daemon_name> is the name of the daemon.

And to stop the tracing use the command:

```
$ /etc/traceit –d <daemon_name>
```

The log files can be found under the directory /tmp/log/<daemon_name>. The log file will contain all the Unix API calls made by the daemon along with the values of arguments passed and return value from them.

Packaging and Deploying Applications

The developer can archive the files using the UWIN utilities like pax or tar and use it to deploy the application at the customer location. In this case, the customer should have access to UWIN utilities (pax or tar) to unpax the archive. This method is suitable when the customer has an installed version of UWIN.

The second option is to bundle the files using an installer package and prepare a setup executable. For the ported applications to work without full version of UWIN being installed at the customer site, the package should be bundled with UWIN Runtime. For more information on UWIN Runtime, please visit the site http://www.wipro.com/uwin

Chapter 3

API-Specific Porting Issues

This chapter deals with the different issues involved at the API level while porting the applications from Unix to Windows NT/9x using UWIN. Note that UWIN supports only X/Open compliant APIs. So, any deviation from the X/Open standard in the application may lead to inconsistencies that are not mentioned here. To overcome these inconsistencies, it is advised to make appropriate changes in the sources so that it is X/Open compliant.

The APIs are categorized into the following topics:

- Process management APIs
- Security APIs
- File management APIs
- Miscellaneous APIs

Process Management APIs

clock()

This function has resolution upto milliseconds only. On Window NT the resolution obtained is 10 milliseconds and on Windows 9x it is 55 milliseconds.

usleep()

usleep() will suspend the execution of the process only till the nearest millisecond.

Security APIs

getpwnam()/getpwuid()/getgrnam()/getgrgid()

In Windows NT, the passwd and group structures are filled with the information from the password and group files generated from the Windows NT user database. In case of Windows 9x, these functions will fail unless the password or group file have been generated manually.

geteuid()/getegid()

In Windows NT, the functions geteuid() and getegid() return a value corresponding to security identifier (SID) as the effective uid or effective gid respectively. In case of Windows 9x, these functions return the uid and gid of the process respectively.

File Management APIs

chdir()

All the windows disk and mapped dries can be accessed from the root directory in UWIN. The drive letter prefixed by a '/' needs to be used for the access.

rename()

Since Windows file systems are not case sensitive, if a file is renamed to a name that is different only in case, it will still be same as the original file.

Miscellaneous APIs

errno / h_errno / perror() / strerror()

UWIN maps Win32 error codes into the corresponding Unix error codes.

getenv() / putenv() / environ

UWIN does not maintain the case-sensitivity of environment variables since on Windows, the environment variable names are not case-sensitive.

Chapter 4

Porting to Windows 9x

In general, the steps to be followed to port an application either to Windows NT or 9x are more or less the same. However there are a few architectural differences between Windows NT and 9x. This chapter will give the developer an insight into the issues on Windows 9x and acts as a guide to sort them out.

Differences between Windows NT and 9x

The differences are categorised into the below sections and discussed in detail under their heading.

- APIs
- Services
- Security
- Error Logging
- Process IDs

APIs

UWIN extensively uses the Win32 APIs supported by the underlying Windows operating system to implement the Unix system calls. Some of the APIs supported by Windows NT are either absent or partially supported in Windows 9x. Hence the behavior of some of the system calls (like security related APIs) will be different in Windows NT and Windows 9x.

Services

In case of Windows 9x there is no concept of services. Hence the /etc/passwd and /etc/group files will not get generated on Windows 9x. However, the developer can have manually generated files.

Security

Windows NT provides the security to the applications that are ported on it. There is no security available for the applications ported on Windows 9x as it does not have the concept of security. Some of the system calls like setuid() becomes a dummy function which just returns success on Windows 9x.

Error Logging

There are totally four log files that are generated in UWIN for different purposes under the directory /tmp where '/' refers to the directory where UWIN is installed. The two log files ums.out and ucs.out will not get generated on Windows 9x since there is no concept of services in it. The other two logs have same meaning as on Windows NT. For more information about the significance of the log files, please refer developer guide Chapter 2: *UWIN Architecture – Error mapping/logging* section.

Process IDs

The process ids on Windows 9x are very huge numbers and hence become negative when converted to pid_t type. Hence UWIN has a complex formula to map the Unix pid to the actual Windows 9x pid.