Portable UPDNODES Installation and User's Guide

March, 1991

This document provides installation and usage instructions for the portable UPDNODES program. UPDNODES is a utility which applies monthly updates to the master nodes database used on BITNET and the cooperating networks.

"Do it yourself and save"

Revision/Update Information:	This is a revised manual.
Operating System and Version:	VAX/VMS V4.4 or later, MS-DOS V3.2 or later, Unix (various), RSTS/E V10.0 or later.
Software Version:	UPDNODES V1.3-04

Academic Computer Center
Saint Peter's College
Jersey City, New Jersey

March, 1991

Permission is granted to copy and redistribute this document provided the copies are made for no commercial advantage.

The information in this document is subject to change without notice and should not be construed as a commitment by Saint Peter's College. Saint Peter's College assumes no responsibility for any errors that may appear in this document.

DISCLAIMER: The software described in this document is provided "**as is**". No guarantee is made by the author or the author's employer as to the suitability, reliability, security, usefulness, or performance of this software.

RSTS/E, VAX, and VMS are trademarks of Digital Equipment Corporation

PostScript is a trademark of Adobe Systems

MS-DOS is a trademark of Microsoft Corporation

Unix is a trademark of AT&T

Copyright ©1991 Terence M. Kennedy

Contents

PREFACE v

CHAPTER 1 INSTALLING PORTABLE UPDNODES 1–1

1.1 REQUIREMENTS 1–1

1.2 FILES IN THE KIT 1–1

1.3 INSTALLATION PROCEDURE - VAX/VMS 1–2

1.4 PRE-INSTALLATION PROCEDURE - UNIX 1–2
1.4.1 Defining oldcc 1–2
1.4.2 Defining SHORT_NAMES 1–2
1.4.3 Defining WEIRD_BIT_ORDER 1–2
1.4.4 Setting the Type of int16u 1–3
1.4.5 Selecting the Proper Makefile 1–3

1.5 INSTALLATION PROCEDURE - UNIX 1–3

1.6 POST-INSTALLATION TESTING - UNIX 1–3
1.6.1 Performance Issues 1–3

1.7 INSTALLATION PROCEDURE - MS-DOS 1–4

1.8 INSTALLATION PROCEDURE - RSTS/E 1–4

CHAPTER 2 USING PORTABLE UPDNODES 2–1

2.1 OBTAINING YOUR INITIAL BITEARN NODES FILE 2–1
2.1.1 From a Nearby BITNET site 2–1
2.1.2 From Your Nearest NETSERV 2–1

Contents

2.2 OBTAINING MONTHLY UPDATES 2–1

2.3 LISTSERV-PUNCH FORMAT 2–2

2.4 INVOKING PORTABLE UPDNODES 2–2
2.4.1 Options 2–2
2.4.1.1 -v Option • 2–2
2.4.1.2 -l# Option • 2–2
2.4.1.3 -? Option • 2–2
2.4.2 File Parameters 2–3

2.5 ERROR MESSAGES 2–3

2.6 REPORTING PROBLEMS AND OBTAINING UPDATES 2–6

INDEX

TABLES

1 Tested Environments v
1–1 Component Files 1–1
2–1 File Name Defaults 2–3

Preface

This manual describes how to install and use the portable UPDNODES program.

The BITEARN NODES file is the master repository of information about all nodes on BITNET and the cooperating networks. As such, various application programs refer to it for information. This file is quite large (as of this writing, it is over 3.5 megabytes). Thus, it is too large to ship to all nodes every month. Instead, a special update file is shipped, giving only the changes made in the previous month. The UPDNODES utility applies this update and generates a new BITEARN NODES file.

Until now, there were only two implementations of UPDNODES - the original implementation by Peter Sylvester and Berthold Pasch, in IBM VS Pascal, and a partial version in portable Pascal by Ed Skochinski. With the advent of non-IBM nodes on BITNET and the availability of tools which use the nodes file (such as PAROUTE) for non-IBM systems, the need for a portable version of UPDNODES became clear.

This manual describes a version of UPDNODES written in portable ANSI C. C was chosen because of the widespread availability of good C compilers. In fact, portable UPDNODES performs quite well on systems such as the IBM PC, so even if your BITNET host cannot run it, you may be able to use it on another system.

Portable UPDNODES has been tested in various environments, as shown in the following table:

Table 1 Tested Environments

Host Processor[1]	Operating System	C Compiler
VAX 8650	VMS V5.4-1	VAX C V3.2
IBM PC	MS-DOS V3.31	Microsoft C V6.0A
Sun 4	SunOS 4.1	Unix C[2]
PDP-11/70	2.11BSD Unix	Unix C[2]
PDP-11/70	RSTS/E V10.0	PDP-11 C V1.1

[1]Should also work on other models in the same product line.

[2]Using the compiler supplied with the operating system.

Intended Audience

This manual is intended for system managers and operators with system privileges. You should be familiar with the operating system on your machine as well as the software linking your system to BITNET. You should also be familiar with various aspects of BITNET node management.

Document Structure

This document consists of two chapters.

Chapter 1	Describes how to install portable UPDNODES.
Chapter 2	Describes using portable UPDNODES.

Related Documents

You can find additional information in the following documents, available from LISTSERV@BITNIC or your local NETSERV:

- UPDATE PROCEDUR - Procedure for updating your host information (BITNET members only)
- NETSERV HELPFILE - Procedure for updating your host information (other networks)
- UPDNODES DESCRIPT - Description of the original UPDNODES program
- NODESFMT DESCRIPT - Description of the contents of BITEARN NODES (current format)
- NEWTAGS DESCRIPT - Description of the contents of BITEARN NODES (forthcoming version)

1 Installing Portable UPDNODES

This chapter describes the procedure for installing portable UPDNODES. Since various operating systems have different methods for compiling and installing programs, you may have to refer to your operating system documentation for detailed instructions.

1.1 Requirements

- The source files for portable UPDNODES (see section 1.2).
- A C compiler.
- Sufficient free disk space to store UPDNODES and the nodes database(s).

1.2 Files in the Kit

The portable UPDNODES kit consists of the following files. If you received the UPDNODES PACKAGE you should have all of these files already.

Table 1–1 Component Files

Filename	Description
CHECKSUM.ASM	Assembly checksum routine for IBM PC's
CHKSUM.C	Checksum computation code
COMPILE.PC	Script file to compile on an IBM PC
COMPILE.RSTS	Script file to compile on RSTS/E
COMPILE.VMS	Script file to compile on VAX/VMS
IO.C	I/O routines
LOG.C	Logging routines
MAKEFILE.2BSD	Makefile for 2.11BSD Unix
MAKEFILE.PC	Makefile for IBM PC
MAKEFILE.SUN	Makefile for SunOS 4.1
MAKEFILE.VMS	Makefile for VAX/VMS
MERGE.C	Merges updates into node file
PARSE.C	Tag parsing code
UPDNODES.C	Main program
UPDNODES.H	Common definitions
UPDNODES.PS	This file - PostScript version
UPDNODES.TXT	This file - ASCII version

1.3 Installation Procedure - VAX/VMS

Place all of the above files in a directory by themselves and issue the DCL command:

```
$ @COMPILE.VMS RETURN
```

You should not receive any errors or warnings during compilation. You may now copy the executable file UPDNODES.EXE to a convenient location on your system. You should probably copy the documentation to wherever you store such files. If space is at a premium you may delete the source files as well as the temporary files produced during the compilation process.

You should now define a symbol to invoke UPDNODES. You may either place the definition in one of your initialization files, such as LOGIN.COM, or you may type it in every time you wish to run UPDNODES. Define the symbol as shown below:

```
$ UPDNODES :== $disk:[directory]UPDNODES.EXE RETURN
```

1.4 Pre-Installation Procedure - Unix

Place all of the above files in a directory by themselves. You will most likely have to edit the UPDNODES.H file to define your Unix variant. Search for the first instance of "unix" in the file. You will need to set the following items for your particular system:

1.4.1 Defining oldcc

If **oldcc** is defined, "classic C" style function prototypes will be used. Additionally, SHORT_NAMES will be defined.

1.4.2 Defining SHORT_NAMES

If **SHORT_NAMES** is defined, variable names will be unique within the first 6 characters, in order to accommodate systems where the linker cannot differentiate between symbols which are not unique within the first 6 characters.

1.4.3 Defining WEIRD_BIT_ORDER

The ANSI standard states that the order of bits within a structure element is implementation-specific. The checksum routines use bit fields and need to know how bits are packed. The most common case has the high bit first and then the low bits. All systems that I've tried UPDNODES on except the Suns use "normal" bit order. If your system doesn't, define **WEIRD_BIT_ORDER**.

1.4.4 Setting the Type of int16u

Since various systems have different word lengths, a *typedef* is used to define the proper format of a 16 bit unsigned integer. For most systems it will be either *unsigned int* (for 16 bit processors) or *unsigned short* (for 32 bit processors).

1.4.5 Selecting the Proper Makefile

Two makefiles are provided, MAKEFILE.2BSD and MAKEFILE.SUN. They are similar but invoke different compiler/linker options. You will probably have to do some minor editing to have a usable makefile. Either rename one of them to makefile or create your own.

1.5 Installation Procedure - Unix

Issue the command:

```
make updnodes RETURN
```

You should not receive any errors or warnings during compilation. You may now copy the executable file updnodes to a convenient location on your system. You should probably copy the documentation to wherever you store such files. If space is at a premium you may delete the source files as well as the temporary files produced during the compilation process.

1.6 Post-Installation Testing - Unix

You should now perform a test update run to verify that portable UPDNODES is working properly on your system. Almost all incompatibility problems manifest in the checksum routines, found in the file CHKSUM.C. The best thing to do if you discover checksum problems is to place debugging printf's in the code and use the sample data given in the CHKSUM ASSEMBLE file (available from LISTSERV@BITNIC) to see where things are going wrong. Usually the problems are due to byte ordering or similar issues.

1.6.1 Performance Issues

At least 35% of the CPU time is spent in the wrotl( ) function. Anything you can do to improve performance there is well worth it. The easiest way out is to recode the function in assembly language (as was done for the MS-DOS implementation). There are two versions of the wrotl( ) function, ifdef'd on **SIMPLE**. The default is the complex version, which runs faster on CISC machines such as the VAX. For most RISC architectures, the difference in execution time with SIMPLE is not enough to make it the default. It is provided mainly as an example of what you'd need to do in assembler.

1.7 Installation Procedure - MS-DOS

Place all of the above files in a directory by themselves. Rename the file COMPILE.PC to COMPILE.BAT and issue the command:

```
COMPILE RETURN
```

You should not receive any errors or warnings during compilation. You may now copy the executable file UPDNODES.EXE to a convenient location on your system. You should probably copy the documentation to wherever you store such files. If space is at a premium you may delete the source files as well as the temporary files produced during the compilation process.

1.8 Installation Procedure - RSTS/E

Place all of the above files in a directory by themselves. Note that RSTS/E will truncate filenames to 6 characters and filetypes to 3 characters. All of the UPDNODES components will still have unique names after such truncation. Issue the DCL command:

```
$ @COMPIL.RST RETURN
```

You should not receive any errors or warnings during compilation. You may now copy the executable file UPDNODES.TSK to a convenient location on your system. You should probably copy the documentation to wherever you store such files. If space is at a premium you may delete the source files as well as the temporary files produced during the compilation process.

You should now define a CCL to invoke UPDNODES. You should place the definition your initialization file, [0,1]START.COM. Define the symbol as shown below:

```
$ DEFINE/COMMAND/SYSTEM UPDNODES- disk:[ppn]UPDNODES.TSK RETURN
```

2 Using Portable UPDNODES

To generate a new master nodes file, you will need a previous version of the master nodes file (BITEARN NODES) and one or more update files, along with the UPDNODES executable.

2.1 Obtaining Your Initial BITEARN NODES File

Obviously, before you can apply monthly updates you'll have to have a file to apply the updates to. Thus, you'll have to get a copy of the current BITEARN NODES file. There are two possible places to get this file, as discussed below.

2.1.1 From a Nearby BITNET site

If you know that one of your nearby BITNET sites has an up-to-date copy of the file, you can request it from them. Since the file is quite large, getting it from a nearby site will reduce network load, and you'll get it faster as well.

2.1.2 From Your Nearest NETSERV

If none of your neighboring sites has a copy of the file, you can request a copy from your nearest NETSERV. You should either already know, or find out now, where the nearest NETSERV is. Using whatever method your operating system uses for interactive SEND commands, send the command:

```
GET BITEARN NODES
```

to your nearest NETSERV. The NETSERV will then transmit the latest version of the file to you. Please be patient as this process may take several days (but you only have to do it once).

2.2 Obtaining Monthly Updates

Updates are issued approximately once a month, generally on the first full weekend. The best way to ensure you'll get the updates is to subscribe to them using NETSERV's AFD (Automatic File Distribution) mechanism. To add yourself to the list of subscribers, send the command:

```
AFD ADD password * NODUPD
```

to your nearest NETSERV.

If your copy of the master nodes file is out of date, you may request "back issues" of the update files from NETSERV with the command:

```
GET VERSyynn NODUPD
```

Note that NETSERV will not have update files more than a year or so old.

2.3 LISTSERV-Punch Format

It is quite likely that the NODES and NODUPD files will be shipped in a format known as LISTSERV-Punch (or **LPUNCH**). An LPUNCH converter is trivial to write, and one is available for VAX/VMS systems from the VMSSERV file server at VMSSERV@FHCRCVAX by sending the command:

```
GET BITNET.PACKAGE
```

Note that there is a period in this filename, not a space as in the other filenames you request from servers.

2.4 Invoking Portable UPDNODES

The general form for invoking portable UPDNODES is:

```
updnodes [options] [[[update_file] input_file] output_file]
```

2.4.1 Options

Various options are available to tune the operation of portable UPDNODES. Normally you would not need to specify any options.

2.4.1.1 -v Option

The **-v** option provides verbose messages while processing the update file. A message is logged for each ADD/REPLACE/DELETE action, as well as for the final checksum validations. An "a" may be appended (as in **-va**) to provide even more output for debugging purposes, or to satisfy the terminally curious.

2.4.1.2 -l# Option

The **-l#** option limits the line length of the output file to # characters. The default is 240, which produces an output file identical to that of the EARN UPDNODES PASCAL program. Since such a file is hard (for humans) to read, an option such as -l80 may be given to produce a terminal-width output file. Such output files should be treated identically by any program which manipulates the node database.

2.4.1.3 -? Option

The **-?** option displays a short help and usage message. It also gives the version of the UPDNODES program. Since question marks are normally globbed on Unix systems, you may specify -h as well.

2.4.2 File Parameters

You may specify the file names to be used in the update run. If you do not specify a file name, UPDNODES will use a default name, as shown in the table below:

Table 2–1 File Name Defaults

File Name[1]	Description
UPDATE.NODUPD	Update file name
BITEARN.NODES	Input file name
BITEARN.NEW	Output file name

[1]Upper case on all systems except Unix, lower case on Unix.

From this, it can be seen that if you invoke portable UPDNODES with a single command line argument of VERSyynn.NODUPD, UPDNODES will apply those updates to the file BITEARN.NODES, producing BITEARN.NEW. Once you are satisfied that the output is correct, you may delete the original BITEARN.NODES and rename BITEARN.NEW to BITEARN.NODES.

2.5 Error Messages

Portable UPDNODES can return several error messages. All such error messages begin with the prefix *** *Error:* and then have a detailed message text. The error messages and explanations are:

```
ADD for node without DEL for node.
Doing auto DEL for node.
```

The update file had an ADD statement for a node that already exists.

```
Attempt to ADD existing node node - ignored.
```

The update file had an ADD statement for a node that already exists.

```
Attempt to DEL nonexistent node node - ignored.
```

The update file had an DEL statement for a node that doesn't exist.

```
Attempt to REP nonexistent node node - ignored.
```

The update file had an REP statement for a node that doesn't exist.

```
Base file version is version, delta is for version.
```

This is not the proper update file for the base file. Make sure you have applied all intervening updates in sequence.

`Checksum for entry` *node* `invalid for original base.`
`Verification:` *dec/hex*`, Calculated:` *dec/hex*

The data found in the original file was not correct, or there is a problem in the UPDNODES checksum logic.

`Checksum for entry` *node* `invalid for output information.`
`Verification:` *dec/hex*`, Calculated:` *dec/hex*

The created output data is not correct, or there is a problem in the UPDNODES checksum logic.

`DEL for` *node*`, but` *node* `non-existent.`

The update file had an DEL statement for a node that doesn't exist.

`Error(s) detected during run.`

Summary message given at end of run when other errors were detected.

`ftell(base) returns` *position*

Program logic error.

`fgetpos(base, &cur) returns` *position*

Program logic error.

`get_base_tag: Ooops! Base line shorter than major tag name.`

Program logic error.

`Input :totcks. tag doesn't match calculated :totcks.`
`Verification:` *dec/hex*`, Calculated:` *dec/hex*

The overall checksum for the input file is incorrect. Possible corrupted file or UPDNODES program error.

`Invalid action` *action* `for` *node*`.`

The action field in the update file was not ADD, DEL, or REP. Possible corrupted update file.

`Invalid line length "`*length*`".`

The user requested a line length that was less than 1 or not a number.

`Line found without tag... Line was:`
`>>>`*text*`<<<`

A line was read which did not contain any tags. Possible corrupted file or UPDNODES program error.

`Major-level tag is too long.`

A node name tag was too long. Possible corrupted file or UPDNODES program error.

`min_find: NULL maj pointer passed.`

Program logic error.

`No :totcks. tag in base` *version* `entry.`

An attempt was made to process a version entry that did not have the required total checksum tag. Possible corrupted file or program logic error.

`No :totcks tag in output` *version* `entry.`

An attempt was made to write a version entry that did not have the required total checksum tag. Possible corrupted file or program logic error.

`No major-level tag in Nodes file.`

UPDNODES could not locate any tags in the input file. Possible corrupted file or UPDNODES program error.

`No resulting VERSnnnn entry.`

The VERSnnnn entry was deleted and no new one was added. Possible corrupt update file.

`This version (`*version*`) is not smart enough.`
`You require at least version` *version*`.`

The update file said it requires a newer release of UPDNODES.

`Too many files specified.`

The user entered more than three file names on the command line.

`Unable to add new tags for` *node* `- all used.`

There were more than 75 tags in a node entry. Recompile the program with the TAGS_IN_NODE definition in UPDNODES.H increased.

```
Unable to open type file "name".
```

UPDNODES was not able to open one of the files.

```
Update file does not contain any known actions.
```

The update file you submitted is not an update file, or is not in the proper format. Possibly it has not been converted from LPUNCH format.

```
Updated :totcks. tag doesn't match calculated output :totcks.
Verification: dec/hex, Calculated: dec/hex
```

The overall checksum for the output file is incorrect. Possible corrupted file or UPDNODES program error.

2.6 Reporting Problems and Obtaining Updates

While portable UPDNODES has been extensively tested, problems can arise due to unforeseen configurations, new versions of related software, or oversights. If you find a problem in portable UPDNODES, please contact the author and report it using one of the following methods:

Network mail:

```
terry@spcvxa.spc.edu              (Internet)
terry@spcvxa.bitnet               (BITNET)
...!rutgers!njin!spcvxb!terry     (UUCP)
```

Regular mail:

```
Saint Peter's College
Academic Computer Center
2641 Kennedy Blvd.
Jersey City, NJ 07306 USA
Attn: Terry Kennedy
```

By telephone:

```
+1 201 435 0252
```

Portable UPDNODES and other tools for node management are discussed on the NODMGT-L network mailing list. Announcements of new versions, bug fixes, etc. are also made via this mailing list. To subscribe, send a network mail message to LISTSERV@BITNIC with the text SUBSCRIBE NODMGT-L *your-real-name*

New versions of portable UPDNODES and other tools are available from LISTSERV@BITNIC (in the TOOLS FILELIST).

Index

A

AFD • 2–1

C

CCL • 1–4

E

Error messages • 2–3

I

int16u • 1–3

L

LISTSERV-Punch • 2–2
Lpunch • 2–2

M

Mailing list • 2–6
makefile • 1–3
Messages
    Error • 2–3
MS-DOS • 1–4

O

oldcc • 1–2
Options • 2–2

P

Performance • 1–3
Problems
    Reporting • 2–6

R

Reporting problems • 2–6
Requirements • 1–1
RSTS/E • 1–4

S

SHORT_NAMES • 1–2

T

Testing • 1–3
typedef • 1–3

U

Unix • 1–2
Updates • 2–6

V

VAX • 1–2
VMS • 1–2

W

WEIRD_BIT_ORDER • 1–2