Installation & Configuration

The installation of uniBasic under Unix is an interactive process. Upon completion, uniBasic and other supplied C utilities are placed into the directory */usr/bin*. A master directory *ub* is created with a system logical unit, containing DCI supplied drivers ($LPT) and system processors (BUILDXF, QUERY, MAKEIN, etc.). Following installation, any user familiar with the IRIS or BITS systems can operate uniBasic and feel quite comfortable. Before you can convert an existing end-user, or install a new system, you will require more Unix knowledge than is provided in this manual.

Configuring Unix for uniBasic

Prior to installation for an end-user, several Unix system parameters may require re-configuration for multi-user operation. This process varies from system to system. When purchasing from a uniBasic Distributor, inquire whether these parameters have been pre-configured for your needs. If changes are required, most systems include a system administrator shell to assist you in necessary reconfiguration. For specific information, contact your manufacturer or distributor before changing any system parameters.

Number of Processes

Each program or command, including login (getty) or a copy of uniBasic, is called a process. Unix maintains a fixed-length table of all active processes on the system. The uniBasic statements **SWAP**, **SPAWN**, and **CALL 98** (phantom port operations) initiate additional processes. Opening a printer may invoke as many as 5 processes temporarily. If the number of system processes is exceeded, an error message may be reported to the console (such as NO MORE PROCESSES for SCO-Unix systems), or uniBasic may generate a negative (system) BASIC error to the application.

To accommodate Windows, **SWAP**, **SPAWN** and print jobs, set the number of processes no less than the number of users * 5. Applications that provide linkage to other Unix applications (such as uniBasic IQ, Word Processing, etc) may require

additional processes per user. The current processes may be displayed using the Unix **ps -ef** command.

Number of Open Files

Unix maintains tables for all opened files on the system. Each process requires a minimum of three (3) channels referred to as: standard input, standard output and standard error. In addition, a process may require additional channels if other files or devices are opened for access. uniBasic itself is an example of a process under Unix. For each additional concurrent process, an additional (3) channels minimum are required.

uniBasic requires a total of 4 channels per user process. These include the standard (3), plus one for the error message file (**ERRMESSAGE**). In addition, each device or data file opened requires one system channel; Indexed files require 2 channels.

See Also: Indexed Data Files

When the configured number of system-wide channels is exceeded, an error message may be reported to the console (such as NFILE for SCO-Unix systems), or the program may generate a negative (system) BASIC error.

To compute the approximate number of channels required for your system, multiply the Number of Processes * 3 to yield the minimum number of channels. Add to that result the average number of opened channels per-user times the number users. Remember to count each Indexed file as two channels, and include provisions for other applications, such as IQ.

Example: An 8-port system with 10 open files per user and 50 processes, might require 300 open files.

Number of Open i-nodes

Unix maintains a table of opened inodes (or header blocks). Each unique file or device opened requires one entry. Ten users accessing the same file typically share the same open i-node.

Number of Locks

Unix maintains a table of read and write locks placed on files by individual processes. Each locked region requires one entry in this table. A locked uniBasic data file record is an example of an entry in the lock table. Indexed file key maintenance temporarily requires several locks for the various levels in the ISAM tree structure. A minimum of 5 locks per process should be adequate for most installations.

Message Queues

For all inter-process communication, uniBasic relies on Unix message queues. Each DCI product creates a message queue at startup to transmit and receive data between users. Such messages include:

- SIGNAL 1 & 2 and SEND/RECV data between ports
- CALL 98 and PORT statement commands and status
- CALL $MONITOR requests and status for PORT ALL MONITOR
- CALL $INPBUF type-ahead data returned to parent process
- MSG command text
- Security communications

On most systems, the Unix command **ipcs** may be used to display information about message queues. Each message queue is identified by a unique 32-bit number, usually displayed as an 8-character hexadecimal value.

DCI products are identified by our own numbering sequence, which when viewed in hexadecimal, take on an appearance such as DC00pnnn. The digits correspond to:

DC **Dynamic Concepts Product**

00 Always zero

p DCI Product ID:
0 Passport daemon
1 uniBasic IRIS
2 uniBasic-N (Niros)
3 uniBasic-IDOS
4 uniBasic IQ

nnn uniBasic port number, in hexadecimal, associated with this queue. For example, port 15 is displayed as "00F".

See also: Terminating a uniBasic process

Message queue requirements for uniBasic are based on the number of concurrent users and overall message traffic on the system. The default values on many systems are sufficient to support a few users, but certainly will need to be increased for large installations. If they are not configured, uniBasic may fail at start-up, possibly with a message such as "Bad system call."

The following 7 parameters affect message queues on most systems. The actual parameter names may vary:

MSGMNI Maximum number of message queues. Configure based upon the maximum number of concurrent uniBasic users plus phantom ports plus uniBasic IQ users plus one for the passport security daemon.

MSGMAX Maximum size of a message in bytes; at least 516.

MSGMNB Maximum number of bytes per message queue. Set to the maximum allowable value; typically 32768.

MSGTQL Maximum number of outstanding system wide messages. Suggested setting is at least 256, but may be adjusted if message activity is known to be greater or smaller.

MSGSSZ Size (in bytes) of a message segment. Memory for message data is divided into segments of the defined size. A value of 32 is recommended.

MSGSEG Number of message segments within the system. MSGSEG * MSGSSZ determines the total number of bytes reserved for message data. The recommended formula is MSGSEG = (MSGTQL * 512)/MSGSSZ. For 256 uniBasic concurrent messages, the value would be: (256 * 512) / 32 = 4096.

MSGMAP Number of entries in the message map table. Each entry represents a contiguous free area in the message segments. The recommended formula is MSGMAP = MSGSEG/8 which, using our example, would be 512. If uniBasic reports "Communication buffer is full" when the actual number of outstanding messages is < MSGTQL, first increase MSGMAP. If that doesn't correct the error, increase MSGSEG.

AIX Note: There are no user-configurable message queue parameters on AIX. The parameters are hard-coded in the kernel, and seem adequate for most installations.

The following points must considered during configuration:

- Free message space must be available on the system. If the queues become full, additional users, including phantom ports, cannot be launched into uniBasic or IQ. In addition, existing users may be prevented from performing **SWAP**, **SPAWN**, **CALL 98** and **PORT** statements, as well as commands such as **PORT ALL MONITOR**.

- A processes queue and any waiting messages are deleted if and when the port exits normally. If a process is killed, it cannot delete its queued messages.

- The configuration guidelines shown above consider only uniBasic requirements. They do not include requirements of other Unix applications which rely on message queues.

Unix Accounting & Protection System

Access to Unix files is regulated by file permissions. Permissions are generally *read*, *write*, and *execute* (other permissions and attributes exist, but are not important for discussion here). These permissions are applied against three levels: The owner/creator of the file, other users in the same group as the creator, and other users in different groups. The permissions are either expressed as letters (rwx) or numbers (4 2 1) added together. When expressed as letters, a nine-character field represents the three levels; numbers are shown as three digits.

Each user gains access to the system through a *login user name* which is assigned to a user number; the *user id*. Normally, no two users share the same *login user name* or *user id*. Each *user id* belongs to a *group*. Group numbers are equated to names in the Unix system file */etc/group*, and *user id* numbers are equated to *login id's* in the file */etc/passwd*.

Creating a Unix Account for uniBasic

Prior to installation, a master (manager) account must be created to own the uniBasic distribution files, programs and directories. Most systems supply a menu-driven administration program to assist with user account management. Please refer to the System Administrator's Guide included with your operating system. Before proceeding, please ensure that the following is completed; bracketed information is user-selectable:

- Create a login id, [unibasic], belonging to a new group, [unibasic], with its own home directory, [/usr/ub].

See also: Configuring a uniBasic Environment

uniBasic Security & Licensing

You may select either Hardware or Software licensing (security) for an installation of DCI software. Both are controlled by the supplied daemon, **/etc/passport**, which is automatically launched by uniBasic. Whereas Software licensing is based upon information derived during installation, Hardware licensing is based upon the external DCI Passport™ device.

In either case, each uniBasic installation is identified by a unique 32-bit license number, generated by the Passport daemon. This license number, along with a DCI supplied Software Selection Number (SSN) activates your installation for various DCI products and configurations.

A license number is expressed as an 8-character hexadecimal value, such as 99D04832. The first two characters represent a specific operating system and/or hardware platform, in this example 99 = SCO Unix, for which the license is granted. Licenses are not transferable to other platforms.

A special directory, */etc/DCI*, is created during installation to maintain security specific information and files for use by all Dynamic Concepts software products. Typically, the following text files are recorded within the directory:

- *ssn* DCI activation key for this installation.
- *osn* OEM activation keys enabling encrypted application software.
- *passport.cmd* Command text used to initiate the Passport daemon.
- *passport.log* Log file maintained by the Passport daemon with security information, licensing methods and errors. During installation and **ubconfig**, you may select the name and location of this file.

If the installation utilizes software security, one additional binary file is created:

- *license* License number information for systems installed with Software licensing.

Warning: Modification, deletion, renaming or moving the *license* file will possibly deactivate a software license number.

Configuration of licensing is initially performed during installation, or by later usage of the supplied **ubconfig** utility.

Software Licensing

Software licensing is based upon information derived from the system by the **/etc/passport** daemon. When launched for the first time, a license file, */etc/DCI/license* is created by the daemon to record the unique license number for this installation. Although several types of software licensing methods are supported, the type is fixed by DCI for each specific hardware and operating system platform. The actual type used on a system is recorded in the Passport Log file.

Software licensing is more fragile than Hardware licensing because the unique 32-bit license number may change due to any number of conditions, including, but not limited to, any of the following:

- Replacement of a disk drive and/or restoration of all data
- Upgrade and/or replacement of the operating system
- Disturbing the */etc/DCI/license* file
- Replacement of a serialized CPU board

In installations where third-party support personnel might disrupt licensing, installation of Hardware licensing is recommended.

Should your system lose it's license, a new license number will be generated automatically. Contact your supplier with your old and new license numbers for a replacement.

Hardware Licensing

Hardware licensing is based upon the connection of a Passport device to an unused serial RS232 communication channel on the computer. Each Passport device is pre-programmed with its own unique 32-bit license number and any given SSN for that license number is perpetual. The Passport device and associated SSN may be installed on another like platform at any time.

For information concerning physical Passport installation and testing, please refer to the documentation supplied with the device.

Loading the Distribution Diskettes & Tapes

The Installation Tape or diskette is first copied into the /*tmp* directory using the supplied floppy or tape and the instructions printed on the media. The name of the device used by DCI to make the tape is listed, i.e. /dev/fd096. In many cases, similar computer systems sold by various manufacturers have differing device names. If you experience difficulty loading the distribution media, contact your supplier to ascertain the exact device name for the format supplied.

For example, SCO Unix distribution is typically 5.25" diskette. The commands to load this type of distribution would be:

```
sign on root
cd /tmp
cpio -iavcdu </dev/fd096
```

Installation media delivered on cartridge tape may have the bytes reversed. If an error occurs during the **cpio** command, try the following command:

```
dd if=[device]  conv=swab | cpio -iavcdu
```

where [*device*] is the device name (i.e. /dev/rtp) shown on the installation tape.

If the command is successful, a list of filenames is displayed as the data is loaded into the /*tmp* directory.

If you have additional installation diskettes/tapes for other DCI products, follow the loading instructions for each, prior to issuing the **ubinstall** command.

Loading the uniBasic Installation Media

Verify that you are signed on as *root*, and defaulted to the /*tmp* directory. Issue the following command to load the Installation media:

```
cpio -iavcdu < device name (i.e. /dev/fd096, /dev/rtp, etc.)
```

The following list of filenames should be printed:

	ub	ub/errmessage
ub/loadlu	ub/lptfilter	ub/makeosn
ub/makesp	ub/passport	ub/sys
ub/sys/batch	ub/sys/buildxf	ub/sys/change
ub/sys/clk	ub/sys/copy	ub/sys/dir
ub/sys/dir1	ub/sys/dokey	ub/sys/dsp
ub/sys/format	ub/sys/keymaint	ub/sys/kill
ub/sys/libr	ub/sys/lpt.bits	ub/sys/lpt.iris
ub/sys/lpt.sample	ub/sys/make	ub/sys/makecmnd
ub/sys/makein	ub/sys/makeitem	ub/sys/mfdel
ub/sys/dsp linked to	ub/sys/pdp	ub/sys/pdphelp
ub/sys/port	ub/sys/query	ub/sys/scan
ub/sys/term	ub/sys/term.ansi	ub/sys/term.tvi925
ub/sys/term.wyse50	ub/sys/term.wyse60	ub/sys/who
ub/ubcompress	ub/ubconfig	ub/ubkill
ub/ubrebuild	ub/ubterm	ub/unibasic
ubinstall		

Loading the uniBasic Development Media

Verify that you are signed on as *root*, and defaulted to the */tmp* directory. Issue the following command to load the Installation media:

```
cpio -iavcdu < [device name] (i.e. /dev/fd096, /dev/rtp, etc.)
```

The following list of filenames should be printed:

ubdev	ubdev/Makefile	ubdev/Release.h
ubdev/call1.c	ubdev/call10.c	ubdev/call105.c
ubdev/call11.c	ubdev/call114.c	ubdev/call120.c
ubdev/call121.c	ubdev/call122.c	ubdev/call123.c
ubdev/call126.c	ubdev/call15.c	ubdev/call18.c
ubdev/call19.c	ubdev/call2.c	ubdev/call20.c
ubdev/call21.c	ubdev/call22.c	ubdev/call23.c
ubdev/call24.c	ubdev/call25.c	ubdev/call27.c
ubdev/call28.c	ubdev/call29.c	ubdev/call3.c
ubdev/call30.c	ubdev/call43.c	ubdev/call44.c
ubdev/call45.c	ubdev/call46.c	ubdev/call47.c
ubdev/call48.c	ubdev/call49.c	ubdev/call5.c
ubdev/call51.c	ubdev/call53.c	ubdev/call56.c
ubdev/call57.c	ubdev/call59.c	ubdev/call60.c
ubdev/call65.c	ubdev/call68.c	ubdev/call7.c
ubdev/call72.c	ubdev/call73.c	ubdev/call76.c
ubdev/call77.c	ubdev/call78.c	ubdev/call81.c
ubdev/call82.c	ubdev/call88.c	ubdev/call96.c
ubdev/call97.c	ubdev/call99.c	ubdev/callavport.c
ubdev/callcimi.c	ubdev/callclu.c	ubdev/calldate.c
ubdev/callenv.c	ubdev/callgbs22.c	ubdev/callgbs24.c
ubdev/callgbs35.c	ubdev/callgbs36.c	ubdev/call37gbs.c
ubdev/callhelp.c	ubdev/callinpbuf.c	ubdev/callmemcmp.c
ubdev/callphil.c	ubdev/callrpcs.c	ubdev/callswapf.c
ubdev/calltrack.c	ubdev/callwindow.c	ubdev/callwlock.c
ubdev/comm	ubdev/comm/comm.h	ubdev/crt.h
ubdev/ctree	ubdev/ctree/ctifil.h	ubdev/ctree/ctport.h
ubdev/decode.h	ubdev/eval.h	ubdev/extern.h
ubdev/files.h	ubdev/math.h	ubdev/misc.h
ubdev/pcode.h	ubdev/read_me	ubdev/runtime.h
ubdev/str.h	ubdev/term0.c	ubdev/term101.h
ubdev/timer.h	ubdev/ubdef.h	ubdev/ubdef1.h
ubdev/ubdef2.h	ubdev/ubdefs.h	ubdev/ubport.o
ubdev/unibasic.o	ubdev/unix.h	ubdev/usercalls.c
ubdev/var.h	ubinstall	

Loading the uniBasic IQ Installation Media

Verify that you are signed on as *root*, and defaulted to the */tmp* directory. Issue the following command to load the Installation media:

```
cpio -iavcdu < [device name] (i.e. /dev/fd096, /dev/rtp, etc.)
```

The following list of filenames should be printed:

iq	iq/bin	iq/bin/ddconvert
iq/bin/ddindex	iq/bin/ddmaint	iq/bin/ddupdate
iq/bin/exportiq	iq/bin/importiq	iq/bin/info2cap
iq/bin/iq	iq/bin/iqconfig	iq/bin/iqmkhelp
iq/bin/maketerm	iq/bin/msgindex	iq/bin/tconvert
iq/dd	iq/dd/ddmaster.dat	iq/dd/ddmaster.idx
iq/dd/inv.dbf	iq/dd/payroll.dta	iq/misc
iq/misc/ddtran.dat	iq/misc/ddtran.idx	iq/misc/iqconfig.dat
iq/misc/iqkwd.dat	iq/misc/iqprfile.def	iq/misc/iqprt.dat
iq/misc/iqscrn.dat	iq/misc/iqscrn.idx	iq/misc/iqtext.dat
iq/misc/iqtran.dat	iq/misc/iqtran.idx	iq/misc/iqtranger.dat
iq/misc/iqtranger.idx	iq/misc/portiq.dat	iq/misc/portiq.idx
ubinstall		

ubinstall - Installing uniBasic Packages

ubinstall is a shell-script designed to run under the *borne* shell only. If the command does not execute immediately, enter the command: **chmod 500 ubinstall** and try starting **./ubinstall** again. If **ubinstall** still fails to begin operation, verify that you are running under the borne shell (usually the file /bin/sh). You can usually start a borne shell by typing **/bin/sh**.

After the desired distribution media is loaded, enter the command:

```
./ubinstall
```

ubinstall will display the following:

```
Installation for "uniBasic" BITS/IRIS Business BASIC emulation

All Rights Reserved.  Copyright (C) 1987 - 1993 by:
Dynamic Concepts Inc.  Mission Viejo, California USA
Installing the following packages:
```

ubinstall will locate all packages loaded for installation. Your display should include one or more of the following packages:

```
   uniBasic BITS/IRIS Business BASIC emulator
   uniBasic Development
   uniBasic IQ

Do you wish to continue? (Yes or No, default = Yes)

Part I)  PASSPORT Security
--------------------------
```

If this is a re-installation of a DCI product, security configuration is omitted and you may proceed to Part II of the installation process. A re-installation is assumed whenever a *passport.cmd* file is located within the */etc/DCI* directory. *Passport.cmd* is the command file that automatically launches the Passport security daemon. If this file already exists, the following message is displayed:

```
Security already configured.  To modify security parameters,
please run the ubconfig utility upon completion of the uniBasic
installation.
```

If this is a new installation, the */etc/DCI* directory is created to store security related files and the following message is displayed:

```
Creating directory "/etc/DCI"...Done
```

Next, the utility **ubconfig** is invoked. **ubconfig** requests only minimal responses to facilitate easy installation.

uniBasic security can be reconfigured at any time after installation by manually executing **ubconfig**, or editing the /etc/DCI/passport.cmd file. The output from **ubconfig** is as follows:

```
UniBasic relies on the background daemon "/etc/passport" for secu-
rity.  The following prompts will guide you through set-up of the
daemon.  These parameters can later be changed by running "ubcon-
fig" or editing the file "/etc/DCI/passport.cmd".

(H)ardware or (S)oftware licensing? (default = "Software")
```

Choose the type of licensing, Software or Hardware, for this installation. When choosing Hardware, verify that the Passport device was properly installed and tested.

When Choosing Software Licensing:

```
Enter the pathname PASSPORT error log:
(default = "/etc/DCI/passport.log")
```

The Passport security daemon periodically records operational information to this log file. Press [RETURN] to select the default log file, or enter the desired full *pathname*.

```
Do you want to append to the log file on each daemon startup?
(otherwise the log is rebuilt) (Yes or No, default = No)
```

Press [RETURN] to append to the log file each time the daemon is started. The daemon is automatically launched by uniBasic, at startup, whenever it is not already running. Rebooting the system or terminating the daemon, following reconfiguration, are examples of restarts which record information to the log file.

Entering NO causes the daemon to re-create the log file during each restart, rather than appending information concerning restarts and errors. For Software Licensing, NO is the recommended response.

```
The following line will be placed in "/etc/DCI/passport.cmd":
```

```
/etc/passport -s </dev/null >/dev/null 2>/etc/DCI/passport.log &

"/etc/DCI/passport.cmd" successfully created.
```

When Choosing Hardware Passport Licensing:

```
Enter the TTY device name to which the PASSPORT is connected:
(Example: /dev/tty1a)
(default = "/dev/null") /dev/tty012
```

Enter the *device name* , i.e. */dev/tty012*, of the serial line connected to the Passport device. The line must be disabled and not in use by any other process.

```
Option: -b baud_rate
Enter the baud rate as selected on the PASSPORT device.
9600, 19200, or 38400:
(default = "19200")
```

Enter the baud rate selected by the switch on the Passport device. The rate may not exceed the capabilities of the device driver for the selected serial line:

```
Enter the pathname PASSPORT error log:
(default = "/etc/DCI/passport.log")
```

The Passport security daemon periodically records operational information to this log file. Press [RETURN] to select the default log file, or enter the desired full *pathname*.

```
Do you want to append to the log file on each
daemon startup? (otherwise the log is rebuilt) (Yes or No, default
= No)
```

Press [RETURN] to append to the log file each time the daemon is started. The daemon is automatically launched by uniBasic, at startup, whenever it is not already running. Rebooting the system or terminating the daemon, following reconfiguration, are examples of restarts which record information to the log file.

Entering NO causes the daemon to re-create the log file during each restart, rather than appending information concerning restarts and errors. Should security errors occur, refer to the **ubconfig** utility for verbose logging.

```
The following line will be placed in "/etc/DCI/passport.cmd":

/etc/passport -b19200 <[tty] >[tty] 2>[logfile] &
```

```
"/etc/DCI/passport.cmd" successfully created.
```

If this is a re-installation, **ubinstall** checks the revision of uniBasic currently installed in */usr/bin*:

```
Checking old unibasic... Level = 5.2

"/usr/bin/unibasic" already exists.  If you install this version,
the current version will be renamed and saved as "/usr/bin/ub5.2".

Do you wish to continue? (Yes or No, default = Yes)
```

A response of NO terminates the installation process. */tmp* will still contain the installation files. All existing uniBasic files and data are unchanged. You may initiate the **ubinstall** operation at a later time without reloading the media. Many systems, however, remove the files in the */tmp* directory whenever the system is shutdown and subsequently restarted.

```
Checking new unibasic... Level = 5.3
```

The next phase assumes you have previously created an account to own the uniBasic distribution files. This master account is the group manager of the uniBasic group, and the owner of the **HOME** directory and *sys* directories (Logical Unit 0) inclusive of all files. Additional utilities placed into the */usr/bin* directory are also owned by uniBasic.

```
Part II)  Accounting Information

uniBasic is distributed with a set of system utilities, an error
message file, sample terminal drivers, printer scripts, etc.
These files have permissions making them generally accessible to
all users, but are installed into the user and group you select.

Enter the user name to receive the distribution files: (default =
"unibasic")
```

Enter the user name previously created as the uniBasic group manager.

Part III) System directory

The system directory is where the distribution files are placed and where the .profile for uniBasic is created or modified. It is normally placed in your account's HOME directory. Other logical units required by your application are best placed in HOME also, unless they should be elsewhere for security or space reasons. The default HOME for new installations is "/usr/ub".

However, the choice of "/usr/ub" is only a default; any directory name on any file system can contain uniBasic logical units, subject to access permissions.

Enter directory to contain system files: (default = "/usr/ub")

If this is a re-installation to the same **HOME** directory (/usr/ub in this case), a warning is printed to avoid overwriting any files or programs normally supplied by DCI that may have been customized by you:

Note: "/usr/ub/sys" already exists.
Installing will overwrite the following files:

attr	batch	buildxf
change	clk	copy
dir	dir1	dokey
dsp	format	iris
keymaint	kill	libr
lpt.bits	lpt.iris	lpt.sample
make	makecmnd	makein
makeitem	mfdel	pdp
pdphlp	port	query
scan	term	term.ansi
term.tvi925	term.wyse50	term.wyse60
who		

If you have made custom modifications to any of these files, you may want to abort the installation at this point and make copies. Otherwise, you can continue and update them to the latest revision.

Do you wish to continue? (Yes or No, default = Yes)

A response of NO terminates the installation process. */tmp* will still contain the installation files. All existing uniBasic files and data are unchanged. You may initiate the **ubinstall** operation at a later time without reloading the media. Many

systems, however, remove the files in the */tmp* directory whenever the system is shutdown and subsequently restarted.

If you are installing uniBasic IQ, you will be prompted as to whether you want to load the *iqdemo* directory. This directory is installed under **HOME** and contains the tutorial files and data dictionaries used in the user manual. It is not necessary to install the tutorial for an end-user system unless you want them to have access to the demonstration files.

```
Do you want to install uniBasic IQ demonstration files and dictio-
nary in /usr/ub/iqdemo ? (y/n) y

Part IV)  Run-time options

Several options in uniBasic are configurable through use of "envi-
ronment variables".  These are generally set up in the file
".profile" in your HOME directory, and are also changeable on-de-
mand when using the shell.  None are required to be set up; de-
faults are used if not specified.
```

BASICMODE Specifies the operating environment for uniBasic. I=IRIS, B=BITS. (default = IRIS).

Select the default emulation mode for users. IRIS mode provides for complete emulation of IRIS commands, syntax and visual operation. CTRL C, Scope mode and Basic modes are enabled. Selecting BITS mode still permits execution and programming of IRIS applications, however command formats are BITS style.

SPC5 Value to be returned by SPC 5 (account number): (default = 65535)

Choose the value to be returned to your programs for this user whenever **SPC 5** function is performed. Since the Unix group, user and protection scheme is numerically different, you are permitted to specify this value rather than have to create a special Unix account number to return your desired value. When different users require different **SPC 5** values, the system is easily changed to test who signed on, and set a different value.

DATESEP Character used to separate MM/DD/YY strings: (default = "/")

Choose the normal date separator used by your applications.

CURRENCY Character used to replace $ in PRINT USING statements: (default = "$")

Select an alternate currency character to be replaced when $ is used in **USING** formats.

WINDOWS Maximum numbers of windows open per user. (default = "0")

If your application uses Dynamic Windows, enter the maximum number of opened windows permitted for each user.

EUROPEAN Mode for date verification calls (CALL 24, 27, 28). 0 = MM/DD/YY, 1 = DD/MM/YY. (default = "0")

For European dates: 31/12/88, choose option 1

You may tailor these Environment Variables as well as a number of other configuration options by later editing the file **HOME/.profile**. For further information on configuration parameters, refer to Configuring a uniBasic Environment.

```
Do you wish to automatically run uniBasic after login? (y/n)
(default = "n") y
```

This configures an automatic launching of uniBasic whenever signing onto an account that executes this standard **HOME/.profile**. You can also specify a BASIC program to start by editing the last line of the .profile script;

See also:: Launching uniBasic from Unix.

```
Installation started: Mon Mar 5 17:42:08 PST 1990
   Creating directory "/usr/ub/sys"
   Creating directory "/usr/ub/dd"
   Installing configuration options in "/usr/ub/.profile"
   Installing file formats in "/etc/magic"
   Installing uniBasic in "/usr/bin"
   Installing distribution files in "/usr/ub/sys
   Creating directory "/usr/ub/ubdev"...Done
   Installing development files in "/usr/ub/ubdev"
   Installing uniBasic IQ files in "/usr/bin"
   Installing IQ screen data files in "/usr/ub/iq"
   Creating directory "/usr/ub/iq"
   Installing demo & dictionary in "/usr/ub/iqdemo"
   Creating directory "/usr/ub/iqdemo"
```

```
Installation completed: Mon Mar 5 17:42:23 PST
```

To run uniBasic, logout ("exit" or "^D") and login to "unibasic".

Finally, the */tmp* directory is cleared. If an error occurs while removing the directory, the following message is printed:

```
There has been an error removing the distribution directory.  Type
<CR> to continue, or Q to quit.
```

Note: To run uniBasic-IQ or ddmaint, there MUST be a terminal definition entry in the /usr/ub/iq/iqcap file for your terminal type. If it does not exist, please type the following command:

```
cd /usr/ub/iq
maketerm
```

If you want to run uniBasic IQ demonstration files, please type the following commands:

```
cd /usr/ub/iqdemo
iq
```

To run uniBasic IQ, logout ("exit" or "^D") and login to "unibasic." Then enter the command "iq".

The installation process has successfully performed the following procedures:

1) Placed the required files in */usr/bin*: unibasic, ubcompress, ubconfig, ubkill, ubrebuild, ubterm, lptfilter, makesp, and makeosn.

2) Placed into *$HOME*: errmessage - uniBasic error message file.

3) Placed into *$HOME/sys*: All system commands, LPT scripts, drivers and terminal control files (term.tvi925, term.ansi, etc).

4) Created the full $HOME/*.profile* environment and startup file.

5) Optionally created the directories *dd*, *ubdev*, and *iqdemo* under $HOME if uniBasic IQ, uniBasic Development or uniBasic IQ Demonstration files were installed respectively. The *dd* directory corresponds to the environment variable IQDD set as a default location to contain your new IQ data dictionaries.

Errors During Installation

If, for some reason, you did not load the files into the */tmp* directory, an error message is printed and you are asked for the actual directory where you loaded the distribution tape:

```
The distribution files cannot be found in "/tmp/ub".  Enter the
location of  the distribution files, or type <CR> to stop the
installation:
```

If you are not logged in from root and attempt to run ubinstall, an error message notifying the user is printed and the installation procedure is aborted.

```
"ubinstall" must be run from the super-user (root) account.

Installation procedure aborted.
```

If you have not created the account to own the uniBasic files, an error is generated and the installation procedure is aborted.

```
You must create an account under which uniBasic can be
installed.  Refer to the System Administrator Guide for your
system.  Most systems have a menu driven program to assist with
account management referred to as the System Administrator
Shell.  This program is known on some systems as "sysadmsh",
"sysadm", "adm", "va", etc. and must be run as super-user
(root).

Installation procedure aborted.
```

If the installation was successful, sign off root, and sign on using the uniBasic master *login id* . Running from root level while performing conversions or building files may render those files protected and inaccessible from other accounts.

Configuring a uniBasic Environment

When you login to Unix, the system typically executes two shell program files. The first is */etc/profile*, owned by root, followed by any optional user *.profile* (dot profile) in the user's **HOME** directory.

The root */etc/profile* usually includes a definition for **PATH**; the directory search path for commands entered at the shell (The system Command Line Processor). It may also contain commands to print a banner, news of the day or mail.

The user's **HOME/***.profile* contains definitions of environment variables, and special commands unique to the particular user signing on to the system. This may include changing the default working directory, and/or automatically launching an application environment such as uniBasic. During **ubinstall**, the *.profile* is modified within the **HOME** directory defining only the required configuration environment variables. The following sections describe configuration options, using Environment Variables, for uniBasic.

All users created with an identical **HOME** directory automatically run the same *.profile* at login. When creating multiple user accounts, you may default all users to the same **HOME** directory, or copy the supplied default *.profile* to each of the users newly established **HOME** directories. Once copied, modify the environment variables (such as **LUST** or **SPC5**) specific to that user accordingly.

During installation, the directory **HOME/sys** is created to contain the *sys* logical unit (0). Other logical units may be created under **HOME**, or in another file system entirely.

Directories and Paths

A Unix file system directory is tree-structured beginning at the level known as *root*. Files are accessed by supplying a *pathname* in the form *dir1/.../filename* through the tree. Since IRIS and BITS applications have been designed for a single level directory, uniBasic provides a Logical Unit Search mechanism to facilitate single to multi-level directory organization. An Environment Variable may be defined specifying the Unix directories to search for Logical Units and/or Packnames. The environment variable named **LUST** (Logical Unit Search Table) in the *.profile* is used to define the paths to the final level directories with unit numbers (or packnames).

Filenames and Pathnames

Filenames are converted to a series of pathnames, appended one at a time to the entries defined by the Environment Variable **LUST** (Logical Unit Search Table) until a match is found. Standard BITS or IRIS filenames are converted to lower case characters; the Unix standard. Filenames beginning with / are assumed to be full Unix names, and no conversion or logical unit search list is performed. The form *pack:file* is converted into pack/file. Account branch characters (%&#, etc) and account [grp-usr] suffixes are discarded. Filenames in the form *0/filename* are converted into s*ys/filename*; other files in the form *lu/filename* remain as is except leading zeros are dropped from the lu number.

Note: An ISAM file is made up of (2) separate files; the lower-case filename holds the data portion and an uppercase filename is created to hold the ISAM portion. Filenames that do not contain at least one letter cannot be used for ISAM data files. See Indexed Data Files.

Organizing Logical Units and Packnames

File access under Unix performs best when directories are balanced, that is kept to a reasonable number of filenames per directory. You may create more than one logical unit (Directory) with the same name to balance your programs and files into a more manageable file structure.

The following illustration shows various ways to organize directories. You simply list all of the paths in the **LUST** variable to your final logical unit or packname directories. A null path (leading or trailing colon) is replaced by your current default working directory.

```
      /(root)                    /(root)
   usr          acct         usr          acct
   ub           progs        files        ub    2 3 4
sys 1 2         ar ap        ar  ap     sys 1
                1  1         2   2
```

The rightmost example shows the simplest structure. Logical Unit zero (sys) and 1 (containing application programs) are under the path */usr/ub*. Data files are on units 2, 3 and 4 under a separate file system (or disk drive) referenced (mounted) as */acct*. The search path for this configuration would be:

```
LUST=:/usr/ub:/acct:/usr/ub/sys
```

In the leftmost example, the sys or LU 0 directory as well as Logical Units 1 and 2 are under */usr/ub*. Both Programs and Files are separated into their own directories (*progs* and *files*) with duplicate logical units 1 and 2 underneath. Assuming all files are accessed as "lu/filename", the appropriate search path for this configuration would be:

```
LUST=:/usr/ub:/acct/progs/ar:/acct/progs/ap:/acct/files/ar:
      /acct/files/ap:/usr/ub/sys
```

In both cases, you may specify paths to a specific directory if your applications do not specify a hard-coded LU. The entry */usr/ub/sys* is normally included as the last entry in **LUST** to force a search of LU 0 when a command is entered; such as **LIBR** or **DIR**.

Other default units can be selected as well, but it is recommended that they be at the end of the **LUST** to minimize searches. Always construct the search paths in a way that minimizes the total number of searches done for each **CHAIN**, **OPEN**, etc.

Environment Variables

This section discusses the user-configurable uniBasic Environment Variables. Definitions are added to, and exported from, a user's *.profile* when the default value is insufficient. It is unnecessary to include definitions when a variable's default value is adequate.

ALTCALL Defines the set of BASIC CALL numbers used within your application that have equivalents as a different number. For example, your application uses CALL 64 to verify date inputs. uniBasic includes a CALL 24 functionally compatible for your requirements. Setting alternate 64=24 invokes a CALL 24 whenever the application requests CALL 64. Multiple CALLs can be defined separated by colons, i.e.: ALTCALL 64=24:62=22 See CALL.

AVAILREC Defines the numeric value to be returned whenever an INDEX / SEARCH Mode 1 requests the number of available records in an ISAM file. If AVAILREC is defined, its value is always returned. When undefined, the number of available records is computed by subtracting the number of active records from the created or current file size. See also: Indexed Data Files.

BASEYEAR Defines the system Base Year to be returned for the function SPC 20. It is also used to compute the hours counter returned for the functions SPC 2 and TIM 2 functions. The default for BASEYEAR is 1980 unless specified in the environment. Because Unix systems maintain clock values beginning in 1970, you may set BASEYEAR to any value from 1970 to the present year. Setting this value outside this range will result in very large (or negative) values for these functions.

BASICMODE Selects the desired operating environment. The default is IRIS emulation with separate SCOPE and BASIC Program command modes. By setting BASICMODE=BITS, you operate in a BITS environment, that is both commands and BASIC statements are performed at a single command prompt.

The NEW command defaults to IRIS or BITS syntax based upon the BASICMODE selected. The NEWI or NEWB commands override the default BASICMODE for creation of new programs. Either BASICMODE runs both types of programs.

Program Files are flagged for IRIS or BITS execution automatically. Text files accessed using LOAD or MERGE take on the type of the current mode. The BITS GET or GETI commands allow you to choose the encoding and runtime format for the text files you access.

BCDVARS If defined and non-zero, all BASIC variables are stored in memory using IRIS BCD format. BCDVARS is required when special CALLs indiscriminately copy data between numeric and string variables by straight memory copy. Do Not set this environment definition without specific instructions from your Distributor or Dynamic Concepts Inc. **See also:** IRIS BCD Files.

BITSPROMPT Change the default prompt * displayed in BITS mode. Format is: BITSPROMPT='replacement string'

CURRENCY Define a single character to be output by **USING** whenever the **$** operator is used. Format is: CURRENCY=replacement character.

DATESEP Define a single character other than '/' to separate MM/DD/YY or DD/MM/YY strings. Format is: DATESEP=replacement character.

DXTDSIZ Specifies the number of records to extend an Indexed file when the data portion is full. The default is 1 record. During creation of an indexed file, this value (or default) is read and stored in the file header. Later expansion of the data portion is based upon this size. Once created, this parameter cannot be changed for a file. Depending on your application, changing this value and **IXTDSIZ** can have some effect on performance.

See also: IXTDSIZ, and Indexed Data Files.

EURINPUT Selects the programming mode used for **USING**. The default (or zero) mode requires programs to use comma and period in the form: #,###.## When set to one, programs use the international form: #.###,##.

See also: USING and EUROUTPUT

EUROPEAN Mode for date input/output formats; 0 for USA Format: MM/DD/YY, 1 for the international format: DD/MM/YY.

EUROUTPUT Selects the output mode for **USING**. Periods and commas are reversed at output. The default (or zero) mode outputs in the format: 1,234.56. When set to one, commas/periods are reversed; output is represented by the form: 1.234,56.

See also: USING and EURINPUT..

GOSUBNEST Selects the maximum number of **GOSUB** and **RETURN** nesting levels in any program. Default is 8 levels deep.

FORNEXTNEST Select the maximum number of **FOR** and **NEXT** nesting levels in any program. Default is 8 levels deep.

IBITSFLAG Set to 1 to eliminate the standard IRIS errors: Channel Already Opened (on **OPEN** Statement), and Selected Channel is not OPEN (**CLOSE** Statement). An **OPEN** issued to an already open channel performs an implied **CLOSE** of the channel first.

INPUTSIZE Size in bytes of the input buffer. This size limits the length of a BASIC statement, **LOAD**, **GET** and other operations, such as Long **CHAIN**, that require the Input buffer.

ISAMBUFS Number of buffers allocated for shared memory ISAM files. This parameter is unused at the time of this writing. DO NOT USE THIS VARIABLE.

ISAMFILES Maximum number of opened Indexed file directories. For each file opened, one entry is required for each Directory (index) plus 1 for the data file. The default value 50 supports 10 indexed files open with an average of (4) directories (indices) each. If this value is too small, the error Illegal Channel (or ISAMFILES value too small) is printed.

ISAMOFFSET Define the displacement within ISAM records for maintenance of a system Deleted Record Flag and Delete Link List pointers. Change this offset (Default 0) when your applications write data within the first 5 bytes of a record following deletion.

See also: Indexed Data Files.

Note: Following deletion of a record, DO NOT WRITE (clear) the entire record or the delete list will be corrupted.

IXTDSIZ Specifies the size in bytes to expand a file's Index portion when the index is full. The default is 512 bytes. During creation of an indexed file, this value is read and stored in the file header. All further access and expansion to the file's index portion is based upon this size. Once a file is created, this parameter cannot be changed for that file unless the file is rebuilt using a new **IXTDSIZ** value. Depending on your application, changing this value along with **DXTDSIZ** (and then rebuilding a file) can have a great effect on performance.

See also: DXTDSIZ

LOCKRETRY Record lock retry counter. A value of zero (default) provides for unlimited record lock return (aka IRIS Revision 7). Any other positive value selects the number of retries (in 5 second intervals) attempted prior to issuing a Record Lock error to the application.

See also: Record Locking

LONGVARS Change the default (0) setting to provide for the global use of long variable names. When set to 1, long variable names are allowed globally; setting 0 disables long names. The **variable** command may be used to override this default at any time.

LUST Logical Unit Search Table. Defines the entire series of Unix paths to search for program and filenames in the form *filename*, *lu*/*filename* or *pack:filename*. If this parameter is not defined, only the current working directory is searched. Filenames beginning with / are assumed to specify the entire path to the file and the **LUST** definition is not used. The following table illustrates the search paths used for a simple *filename* and *lu*/*filename*.

```
LUST=:/usr/ub:/ub/sys:/usr/ub/1:/usr/acct:/usr/acct/2
```

filename	pack:file or lu/file
filename	lu/filename
/usr/ub/filename	/usr/ub/lu/filename
/usr/ub/sys/filename	/usr/ub/sys/lu/filename
/usr/ub/1/filename	/usr/ub/1/lu/filename
/usr/acct/filename	/usr/acct/lu/filename
/usr/acct/2/filename	/usr/acct/2/lu/filename

LUST should be constructed to minimize number of searches required to locate programs and files. If an application under IRIS or BITS defaults to a specific logical unit containing programs or data, set the current working directory to that same location. This is accomplished by including a **cd** *pathname* command within the *.profile*.

Note: The maximum number of entries in the **LUST** is 24.

If all file and program access is in the form *lu*/*filename*, or *pack:filename*, define **LUST** to provide the path to the directory containing the actual numbered (or named) logical units only.

If you rely on the IRIS or BITS LU Search for other Logical units, then you must include full paths directly to each directory.

To ensure the fastest access to programs and files, determine whether your application performs more **OPEN** or **CHAIN** statements. List your entries in **LUST** accordingly. If most filenames include a Logical Unit or packname, list entries terminating at the directory containing the *lu*, and finally list direct paths to each named or numbered directory.

MAXACCSLEEP Define accuracy vs. performance of the Unix sleep timers utilized by **PAUSE**, **SIGNAL 3**, **INPUT TIM**, record locking, etc. Since many Unix systems provide timer accuracy only to the nearest second, uniBasic employs the following software method to ensure accurate tenth-second timers:

First, the specified delay is rounded down to the nearest whole second. If at least one-second of delay is warranted, the process sleeps, allowing other processes to run, for that number of seconds. Following the sleep period, uniBasic 'spins', i.e. wastes CPU time, by watching the clock for the remaining partial second.

Most applications are not timing critical. Substantial system wide performance is realized by configuring delays to round up to the nearest whole second. That is, a delay of 5 tenth-seconds is rounded to a full second.

MAXACCSLEEP defines the delay value, below which, exact accuracy is required. Delays at or above this value always round to the next whole second. A value of zero, the default, provides the highest accuracy at the expense of additional system overhead. A value of one always rounds, etc. To ensure accuracy on all delays below two seconds, set the value to 20.

Some systems support highly accurate timers without the requirement to waste CPU time, including SCO Unix, NCR Tower 7xx/8xx, MIPS and Motorola 88000. These systems default, automatically, to 65535 which enables the system specific timer. On systems that do not support accurate timers, the value defaults to zero.

MAXPORT Change the default automatic *port number* assignment to a value other than 999. The maximum port number is 1023. Used for automatic *port number* assignment by the **SPAWN** statement, and during sign on when **PORT** and/or **PORTS** are undefined. Set to 99 to prevent automatic assignment of 3-digit *port numbers*.

See also: PORT, **PORTS** and Port Numbering and Phantom Ports.

MAXVARS Limit the maximum number of variable names that can be used within a program. 348 unique variable names may be used within each program. Setting **MAXVARS** to 93 ensures backwards compatibility to IRIS or BITS. This value is only checked whenever a program statement is entered adding a new variable to a program.

MSC7 Define the numeric value to be returned by the **MSC(7)** function. If **MSC7** is not defined (or defined as 65535), your UNIX group number * 256 plus the user number is returned. **MSC(7)** will yield unpredictable results when the group or user numbers are greater than 255.

PFCHAR Define a single replacement character for any @ character terminating a *filename*. Format is: **PFCHAR**= replacement character. When **PFCHAR** is not defined, the trailing @ character is ignored and terminates a *filename*. Therefore, the filenames DATA and DATA@ both select the same *filename*. This default operation is recommended as a method of preventing @ characters from becoming part of a Unix filename. @ is not a portable filename character, and its use may interfere with some Unix shell commands.

On some IRIS systems, users may have nearly identical files, such as DATA and DATA@. Defining this option removes the requirement to modify applications and *filenames*.

To define this option, choose a single character to replace @, such as **PFCHAR='-'**. In this example, any attempt to **BUILD** or **OPEN** a filename such as DATA@, results in an operation to DATA-.

Note: This option should only be utilized on systems where a blind conversion is being performed. It will safeguard against conversion errors when an IRIS system has nearly identical data and poly *filenames*. Resellers converting known systems are advised to rename or delete conflicting filenames. Most often, duplications are the result of an older Indexed file (itself no longer in use) being recreated as a Polyfile.

Once files have been built with this substitution in effect, the option must remain set, or all program occurrences of the @ must be changed to the specified replacement character.

See also: Setting up .profile for Multiple Users.

PORT Force the current session to operate as a specific **PORT** number, i.e. PORT=23. The maximum Port number is typically 1023, unless your system is licensed for a greater number of users.

PORTS Define a specific port numbering order. The format of this definition is:

```
PORTS=tty00:tty1b:#7:tty1c . . .
```

In this example, Port 0 is tty00, Port 1 is tty1b; starting at Port 7 is tty1c leaving ports 2-6 unused. When neither **PORT** nor **PORTS** is defined in the environment, *port numbers* are assigned based upon the tty name (tty23 is port 23). If a name conflicts with an existing port (or a port already in use), a number is assigned backwards starting at **MAXPORT**. To prevent automatic assignment, all system tty device names not in the form ttynnn (where 'nnn' are digits) should be listed. Ports conforming to the normal numbering conventions need not be defined.

See also: Port Numbering and Phantom Ports.

PREALLOCATE This variable contains several flags which, when added together, define options for processing data files.

Options fall into two categories, runtime and permanent. Permanent options are indicated by •. Runtime options affect current file operations when enabled. Permanent options affect all future access to files created when that option was enabled. Permanent options are stored within the file's header and typically define file limits or data storage formats.

1 Preallocate all blocks for contiguous files and initialize to zero bytes. You might set this value on a new system to force files to occupy physically contiguous space on the disk. Note: Indexed files store keys in a separate file, and may be built too large using older style IRIS or BITS creation algorithms. If this flag is set, modify your file creation sizing algorithms. Runtime option.

2 Do not allow writing past the original created size of a contiguous file (no expansion). Runtime option.

4 When expanding a contiguous file, do not fill in all records between current end of file and new record to write. ** Runtime option.

See also: Contiguous Files.

8 Check Formatted files and return a Record Not Written error if a record has never been written or contains only null (zero) bytes. Runtime option.

16 When expanding a Formatted file, do not fill in all records between current end of file and new record to write. ** Runtime option.

See also: Formatted Files.

• 32 Always **BUILD** and **CREATE** new files in IRIS style BCD record format. This flag may be required if: a) data files were converted from IRIS and b) your application indiscriminately copies entire records from one file to another using variables other than the actual field specification. For example, a **MAT READ** of a string or 1% array. Setting this flag for new installations forces creation of potentially transportable data records for future relocation to other hardware platforms. Permanent option for files created while enabled.

See also: IRIS BCD Files.

Note: DO NOT set this mode during IRIS or BITS conversions.

• 64 Always **BUILD** and **CREATE** indexed files in IRIS/BITS 8-bit key format. Forces keys to be stored in exact IRIS/BITS format. This flag is required when applications utilize binary information in the keys. DO NOT set this mode during conversion of files from IRIS or BITS. Permanent option for files created while enabled.

See also: IRIS BCD Files and Indexed Data Files.

Note: DO NOT set this mode during IRIS or BITS conversions.

---

128 Restrict Indexed files from dynamic expansion. When built, the number of records specified to **BUILD** or **CREATE** is retained in the file header as the maximum number of records for the file. The status **E=3** is returned from **SEARCH #** and **INDEX#** when the file dynamically expands to this record number. Runtime option.

See also: Indexed Data Files.

256 During Indexed File Record Deletion, check for record already deleted. When deleting records and adding them to the delete chain, this runtime flag forces an initial check of the delete flag prior to deletion. If the record is already flagged as deleted, an exception status (E=1) is returned, and the record is not added to the deleted record list. This flag may be required if your applications arbitrarily delete records not currently in use. Runtime option.

512 Permit writing past the record boundary of an Indexed file in a single operation. Normally, error 144 is generated whenever a single write operation will cross a record boundary. This option should only be used when the application is certain that all records to be written are previously allocated, otherwise the file's deleted record list might be corrupted. This option is runtime in nature, affecting all open Indexed files. Runtime option.

• 1024 Always **BUILD** and **CREATE** new files in IMS style BCD record format. This flag may be required if: a) data files were converted from IMS and b) your application indiscriminately copies entire records from one file to another using variables other than the actual field specification. Permanent option.

Note: DO NOT set this mode during conversion of files from IRIS or BITS.

See also: IMS BCD Files.

2048 Reserved for future use. DO NOT enable this option within your application.

4096 Prevent all write operations to deleted records within Indexed files. Prior to each write operation, the record's delete flag is checked. If the record is flagged as deleted, set **ERR(8)** c-tree status to 144 and return BASIC error 123. Runtime option.

Note: Formatted and Contiguous, including Indexed, files are typically created containing a 512-byte header and no data records. For Contiguous and Indexed files, the number of records specified to **BUILD** or **CREATE** is stored within the header for use by **CHF** and runtime-limiting **PREALLOCATE** options. Only when **PREALLOCATE** option 1 is set are records physically allocated at creation.

Prior to each write operation, the number of records between the current physical end-of-file and the end of the record being written is computed. Missing (intervening) records are automatically written to the file. This process may take several seconds depending upon the number of intervening records that must be written.

When setting **PREALLOCATE** to prevent intervening record allocation, only the record to be written is allocated. Reading any nonexistent record results in the transfer of a null data without error. Although these files are completely valid, warning messages may be printed by the Unix command **fsck** (File System Check) when 'gaps' are detected in the structure. These files are sometimes referred to as sparse files.

Within Formatted files, **PREALLOCATE** option 4 is used to interpret null records as Records not written.

SCOPEPROMPT Choose an alternate prompt while in SCOPE Command Mode (**BASICMODE=IRIS** only). The default prompt # is replaced using the form: **SCOPEPROMPT='**replacement characters'.

SPC5 Define the numeric value to be returned whenever the **SPC(5)** function is called. If **SPC5** is not defined as an environment variable (or set to 65535), your UNIX group number * 256 plus the user number is returned. The **SPC(5)** function will yield unpredictable results when the group or user numbers are greater than 255.

See also: Setting up .profile for Multiple Users.

SPC7 Define the numeric value to be returned whenever the **SPC(7)** function is used. If **SPC7** is not defined as an environmental variable, zero (0) is returned.

STRING Select alternate string processing for BASIC to match HAGEN Business Basic. To invoke HAGEN String Processing, use the form: STRING=HAGEN.

TABFIELD Change the number of spaces between comma fields in **PRINT** statements from 20 to the new numeric value specified.

WINDOWS Define the maximum number of Windows that may be opened by this user. If **WINDOWS** is defined, the main screen is counted as the first Window. Each WINDOW requires approximately 64 bytes of storage for the array. As Windows are created, memory is allocated based upon twice the number of characters in the Window. The main screen occupies (80 *24 *2) characters of memory for a 80 column, 24 row screen.

See also: Windows and Output Considerations, WINDOW, CALL $WINDOW, and MSC Functions.

WARNING: THE FOLLOWING UNIX ENVIRONMENT VARIABLES MAY BE EXAMINED OR CHANGED AS REQUIRED. HOWEVER, CHANGING THESE VARIABLES WILL LIKELY AFFECT THE OPERATION OF OTHER UNIX APPLICATIONS.

HOME The home directory of the user, i.e. */usr/ub*.

HZ The clock rate used internally by the Unix system. For most systems, this value is either pre-defined to the compiler or is already in the environment. This value is used to compute certain **TIM** and **SPC** functions; the **BYE** command and pause durations less than 1 second. Do not change this variable unless incorrect times are reported by the above noted functions.

See also: MAXACCSLEEP Environment Variable.

TERM Many applications, including uniBasic, retrieve the value of this variable to select a terminal driver for screen operations. While many applications rely on the Unix **termcap** or **terminfo** drivers, uniBasic developers have the flexibility of their own driver system.

See also: Configuring Terminal Drivers

PATH The Logical search path for Unix commands issued to the shell. PATH=:path:path:path: ... The **PATH** is only referenced when shell commands (or Unix commands) are entered while in *command mode*. To open pipes without supplying the full pathname (i.e. **DUMP $more**), append **PATH** definitions to **LUST**, i.e.:
LUST=$LUST:$PATH

Note: The following are useful Unix commands that may be of interest to the user. For more detailed information, consult your Unix documentation.

stty Command to reset terminal configuration, Baud rate, parity, backspace and control characters, xon/xoff protocol, character length, mapping of return to return-linefeed, etc.

Unix typically assigns the characters BREAK and DELETE for QUIT and INTERRUPT functions used to abort a process. These functions are reset upon entry to uniBasic to the characters ^D **[EOBC]** and **ESCAPE**.

When a Unix command is performed from uniBasic (Command mode, **SYSTEM** statement), the functions are reset to their initial Unix definitions for the duration of the system command. Some users find it desirable to use ESC and ^D for both system and uniBasic commands. The stty command may be executed from within the *.profile* to change the default Interrupt and Quit functions.

Note: To ensure proper terminal operation, incoming **stty** parameters are saved whenever a uniBasic process is launched. Issuing **stty** or similar commands, within uniBasic, have little effect since uniBasic restores and resets these parameters. Certain changes are permitted, using the **!** command, such as changing the baud rate.

---

cd $HOME/1 Command within *.profile* to set the user's default Logical Unit to 1 when LU 1 directory is below **HOME**.

umask Set to zero to provide for pass through protections to Unix. Any non-zero value forces Unix to XOR supplied protection digits with this umask value. For example, if umask=7, then all lower protection digits are cleared. See File Attributes, Protection and Permissions for a complete discussion of the Unix protection system.

ulimit The ulimit command sets the upper limit (in blocks) for files created on the system. Set this value to the largest allowed value to allow your applications to control file size. If this value is set too low, a Write Error will be given when a file reaches this maximum size. This value may be defined in */etc/profile*, as part of the user's account or within the

Kernel. Contact your supplier if this value is too small for your needs.

Setting up .profile for Multiple Users

When multiple users default to the same **HOME** directory, you may insert statements within *.profile* to determine the *login name* used, and configure environment variables accordingly. The following statements might be added to **HOME**/*.profile*.

If you are running under SCO-Unix, insert the following lines to determine the name of the user signing on:

```
LOGNAME= `who am i`        #2+ users with same id.
LOGNAME= `id`              # unique id names for each user.
LOGNAME= `expr "$LOGNAME" : '\([^]*\)'`
export LOGNAME
```

To set a different **SPC5, MSC7** or **LUST** (Logical Unit search path) based upon the user signing on:

```
case  $LOGNAME in
   "doug")  SPC5=32774;LUST=$LUST:/usr/drive1;;
   "laura") SPC5=32896;LUST=$LUST:/usr/drive2;;
   "mike")  SPC5=32768;LUST=$LUST:/usr/drive3;;
   *)       SPC5=16384;;      #Default other users.
   esac
```

The previous example configures different **SPC5** values and alters **LUST**, appending to its previously defined value the additional search of *drive1*, *drive2*, or *drive3* only for doug, laura or mike. By appending a previous base value, it is unnecessary to redefine the entire **LUST** specification for each user. A total re-definition would take the form:

```
LUST=/usr/ub:/usr/ub/sys:/usr/drive1.
```

For further information , refer to the Unix manuals on Shell Programming.

Command Line Interpreter

Two separate command line interfaces are provided within a running uniBasic process. *Command Mode* is signified by the prompt character **#** (**SCOPEPROMPT**) printed at the left margin. System commands (uniBasic or Unix) and program names may be entered while in Command Mode.

BASIC Program Mode is entered by the **BASIC** Command and has no prompt character. Programming and debugging is performed while in BASIC Program Mode.

```
#ls                Issue Unix Directory command
#LIBR {param}      Command Mode example
READ  var.list     BASIC Program Mode example
```

It is also possible to configure all commands for operation from a single *command mode* by setting **BASICMODE**=BITS. In this configuration, a single prompt ***** (**BITSPROMPT**) is always displayed at the left margin.

```
*ls                Issue Unix Directory command
*LIBR {param}      Command Mode example
*READ  var.list    BASIC Program Mode example
```

Launching uniBasic From Unix

SYNOPSIS: Launch a uniBasic Process
unibasic {-*ffilename*} {-*Ffilename*} {-s} {-o} {-t}

DESCRIPTION

Start a uniBasic session on your terminal. The current environment is read for all pertinent variables, a Port Number is established, a Message Queue is created and the terminal modes are reconfigured. If this is to be an interactive keyboard session, the terminal is placed into *command mode*.

filename is an optional name of any BASIC program file. The specified *filename* must be in the current working directory, or in one of the supplied *pathnames* specified in the environment variable **LUST**. The *filename* may also include a *lu* identifier, or be a full Unix pathname beginning with '/'.

The *-f* switch is used to immediately execute the named *program* file. If the specified *program* terminates or an error occurs, the terminal remains within uniBasic in command mode.

The *-F*switch is also used to immediately execute the named *program* file. However, if the specified *program* terminates using **STOP**, **BYE**, **SYSTEM 0**, **END**, **CHAIN** "", non-trapped **ESC**, **[EOBC]** (CTRL D) or an abortive error, the session is terminated, and control returns to the point uniBasic was launched; see below.

The *-s* switch requests entry of a new Software Selection Number (SSN). The SSN might be changed when you are installing additional terminals, installing additional products (such as uniBasic IQ) or converting a demonstration License into a paid-up License of uniBasic.

The *-o* switch requests the entry of a new OEM Selection Number (OSN). The OSN is used to control execution of one or more dealer-protected software packages.

The *-t* switch requests the entry of a new OEM Selection Number (OSN) similar to the *-o* switch. This OSN is considered temporary and is not stored into the system. The *-t* option is used when the owner of protected software wants to temporarily grant access to the source code. This access is restricted to the single terminal issuing the *-t* switch.

When a session terminates using **BYE**, **SYSTEM 0** or **1**, or an aborting condition using the *-Ffilename*, the process is exited, and all terminal characteristics are reset to the incoming values. If the uniBasic session was started from the shell,

then the shell is resumed. If launched from the *.profile* using a **unibasic {switches}** command, the *.profile* resumes at the following statement. To return the user to login mode at process termination, place an **exec unibasic {switches}** command as the last line of the *.profile*.

EXAMPLES

```
unibasic -f menu
unibasic -s
unibasic -F program | tee savefile
```

ERRORS

No SSN currently entered
Demonstration system, not for resale
License number from ssn does not match actual license
Cannot allocate sufficient memory
Cannot initialize ISAM. Check ISAMBUFS/ISAMFILES definitions
Cannot open term.xxx file. No CRT translation in effect!
Error loading CRT file term.xxxx. No CRT translation in effect!
Could not open 'errmessage', no error messages available!
Too many users; max = n
Port n is already signed on and in use

See also: Environment Variables, Entering an SSN, PORT, PORTS, CRT TERM Files, Program Files, Port Numbering

---

Terminating a uniBasic process

Once initiated, an interactive uniBasic process remains active until terminated. Interactive, as well as Phantom Port, termination is provided for with the **SYSTEM 0** and **BYE** commands.

Non-interactive uniBasic processes, such as those launched using unibasic -F or **SPAWN**, terminate when the specified program stops execution.

All of the above (normal) methods provide for a graceful termination of uniBasic. Open files and devices are closed, the Message Queue is removed, the terminal driver is reset to the modes present upon entry and the process terminates.

Abnormal termination, resulting from the following events, may require operator intervention before other tasks may be performed:

- Memory Fault - core dump
- Hardware failures.
- Receipt of a non-supported signal. uniBasic supports the signals HANGUP(1), TERMINATE(15), SYSCHILD, SIGPIPE, INT, QUIT, SIGUSR1, SIGUSR2. Any other signal may cause abnormal termination.

The following functions may be performed manually, from the failing terminal, when an orderly shutdown did not occur. From a remote location, only the Message Queue must be deleted, after which you should kill any remaining processes, including the shell, associated with the port.

- Issue the Unix command: **stty sane** and press CTRL J or RETURN if the terminal is misbehaving.
- Issue the Unix command: **ipcs** to review, and **ipcrm** to remove the Message Queue for the offending port.
- Issue the Unix command: **ps** to determine and kill any remaining suspect processes under the port's control.
- Sign off and back on to reset all terminal parameters before relaunching another uniBasic process.

See Also: Message Queues

Licensing a New Installation

If this a new installation, you may be asked to enter an SSN the first time uniBasic is launched:

```
$ unibasic

UniBasic Level 5.3
All Rights Reserved.  Copyright (C) 1987 - 1993 by:
Dynamic Concepts Inc.  Mission Viejo, California USA

No SSN currently entered

Enter Software Selection Number (SSN), RETURN to remain the same
```

If you do not yet have an SSN, press [RETURN] to invoke a single-user grace period. A special warning about the grace period is printed periodically until you enter an authorized SSN.

To obtain an authorized SSN for this installation, contact your supplier with the following information:

- License Number displayed
- Number of ports desired
- Type of system
- End-User name
- Options, such as uniBasic IQ runtime, IQ development or IMT.

SSN entry is space and case insensitive. After entering all characters, press [RETURN]. You will be prompted to enter the User Name. Enter the name exactly as printed on the SSN License Agreement. Entry of the name is case and space sensitive. Backspace may be used to correct input errors.

Following entry of the SSN and User Name, immediately issue a **BYE** command, and restart uniBasic. If the SSN was accepted, the *command mode* prompt is displayed. If you are again asked to enter an SSN, either an error occurred during entry, or the License Number does not match the supplied SSN Report.

The SSN contains the licensed configuration for the specific License Number. Currently, an SSN includes Demonstration options (Permits operation for up to 90 days), the number of concurrent Ports that may run uniBasic, and additional information to enable uniBasic IMT and uniBasic IQ.

Note: When using Software Licensing, the license number is keyed to your specific system. Prior to updating the operating system (Unix), or replacing or re-formatting your disk drive, contact your distributor or Dynamic Concepts concerning the deactivation and replacement policy for your license.

Changing the SSN Activation Key

Prior to changing a system's SSN, verify that you have a copy of the existing SSN number, as contained within the file */etc/DCI/ssn*. Prior to installing a new **SSN**, you may print this text file, or use the Unix **cp** command to make a copy of this file. You will need root permission to access this special file.

To change an existing SSN, for example to add additional users, enable additional products or convert a demonstration license into a full license, issue the command:

```
unibasic -s
```

Any existing SSN is displayed.

Enter the new SSN (case and space insensitive) and Customer Name (space and case sensitive). After pressing return, *command mode* is entered.

Following entry of the SSN and User Name, immediately issue a **BYE** command, and restart uniBasic. If the SSN was accepted, the *command mode* prompt is displayed. If you are again asked to enter an SSN, either an error occurred during entry, or the License Number does not match the supplied SSN Report.

See also: Launching uniBasic from Unix.

Launching UniBasic Ports at Startup

You may provide for turn-key operation whereby Unix automatically launches terminals directly into uniBasic, and/or your application. Start-up is performed at system initialization (IPL) or whenever a terminal is evicted or a user signs off.

This feature may be used for interactive or phantom (background) jobs.

The following instructions apply to most Unix based non-server environments. Instructions for Xenix systems are included after this discussion.

Make the following changes for each port to be initialized:

1. When starting an interactive terminal, change the **getty** command inside the */etc/inittab* entry for the terminal to:

```
login unibasic </dev/ttyxx >/dev/ttyxx 2>&1
```

   where 'xx' is the system tty name.

2. Change the *.profile* to set the necessary tty options. The **PORTS** environment variable should be defined within *.profile* to ensure the same *port number* assignment for each automatic startup.

a. *.profile* based upon a Login User Id: Create a login 'ubauto' with the same **$HOME** directory, group and user id as your 'unibasic' login. Then add a single line in *.profile* to handle all automatic startup ports:

```
[ $LOGNAME = ubauto ] && stty sane
```

---or---

b. *.profile* based upon which tty when ports require different settings:

```
case `tty` in
      *tty01)     stty 9600 sane ;;
      *tty02)     stty 1200 sane ;;
esac
```

3. When starting a *phantom port*, change the command to:

```
PORT=n login unibasic </dev/null >/dev/null 2>&1
```

where 'n' is the desired *port number* for the process. No changes are required to *.profile*. You may also include **PORT=n** for interactive ports when the **PORTS** environment variable is not defined, or special numbering for each process is desired.

The 'login unibasic' forces a direct login and execution of the *.profile* as if the login id 'unibasic' was entered on a terminal.

The *.profile* must contain the line **exec unibasic** as the last line to launch the session. The initial copyright is printed and the session is waiting input at *command mode*. You may also force a starting program using the form **exec unibasic -f program.**

For Xenix systems, make the following changes for each port to be initialized:

1. When starting an interactive terminal, issue the Xenix **disable** command for each tty to be started.

2. Create an */etc/gettydefs* entry using a copy of an existing entry with the desired settings. Append # auto to the end of the entry.

3. Change */etc/ttys* to reference the new */etc/gettydefs* entry.

4. Create a login using the tty name, i.e. tty01, which has the same **$HOME** directory, group and user id as your unibasic login. No modifications to *.profile* should be required.

See also: Setting up .profile For Multiple Users, **PORT**, **PORTS**, Port Numbering and Phantom Ports, Launching uniBasic from Unix, Port Number

---

Configuring Printer Drivers

Two printer drivers are supplied for use with your applications; **lpt.iris** and **lpt.bits**. An additional file **lpt.sample** documents various modifications and sample printer drivers.

lpt.iris is designed for applications requiring locked printers. Users attempting access to a locked device receive an error until it is available.

lpt.bits is designed for multi-user spooling applications. Both drivers are similar and may be used with either IRIS or BITS applications.

You may examine and change the driver saving copies using the *filenames* required by the application, i.e. lpt1, lpt2, etc. A driver must use a lower-case *filename* and be stored within a directory listed in the **LUST** Logical Unit Search Table. Do not place a $ as the first character of the *filename*. The $ is a flag recognized by uniBasic as a request to open a *pipe* to an executable file.

For a printer driver to operate correctly, it should be owned by the master uniBasic account with the permissions 555. Before using the driver, issue the Unix command: **chmod 555 *filename*** to set the proper permissions. If further modifications are necessary, issue **chmod 666 *filename***, perform editing as required and reset the permissions to **555**.

The following is a line by line description of the supplied **lpt.iris** printer script. It is designed to run as an executable shell-script under the borne shell only. It operates as a *pipe*, taking as its standard input data transmitted by **PRINT #** statements.

```
#lock LPT - Printer Driver for UniBasic
```

If the first line begins with '#lock', locking is employed to guarantee single user access to the device. Typically required for check or form printers.

Note: No tabs, spaces, blank lines or other characters may exist before the '#lock'.

```
#     Module: lpt Level: 1.2 Modified: 7/18/88
```

Comment indicating revision of supplied lpt script.

```
trap "" 1 2 3
INODE=`ls -i $0`
INODE=`expr "$INODE" : ' *\([0-9]*\)'`
LOCKFILE=/tmp/lk.$INODE
trap "rm $LOCKFILE" 0
```

Setup for cancellation, and signals. Determine the filename of the lock file built, and setup to remove the lock file on script termination.

```
OPENSTR='\c'
```

Define the string of characters to be sent to the printer when opened. The \c is a special flag for the Unix **echo** command to avoid sending a return and line-feed following the characters. Enclose within single quotes; characters as themselves, \0nn for octal using 7-bit form, such as \015 for carriage return; \? special characters such as \n new-line, \r return, \f form-feed. For a complete list, refer to your Unix documentation on the **echo** command.

```
CLOSSTR='\f\c'
```

Define the string of characters to be sent when all output is complete. The same rules apply as with OPENSTR.

```
FILTER='lptfilter BX \010'
```

Define output filtering. Supplied by Dynamic Concepts, **lptfilter** provides output translation. Modify the data between quotes to contain 'lptfilter' and pairs of parameters representing data sent by the application, and replacement strings. The above example changes all BX mnemonics (Begin Expanded Print) to the replacement string ASCII character 10 (octal). For additional information, see also **lptfilter**. **lptfilter** prints directions for its use when typed as a command at *command mode*, or at the shell.

```
PTRDEV='/dev/lp00'
```

Define the device to actually receive the finalized data sent by this script. To send the data through the spooler, this line would contain the actual spool command within single quotes, such as **lp -s**.

```
PTRBAUD='9600 opost onlcr istrip ixon cs8 -parenb'
```

Define for a serial port the baud rate and other characteristics required to define the port for the printer. The above options indicate 9600 baud, process post output, change new-line to carriage return, strip high bit, Xon/Xoff protocol, etc. This string is not required for parallel printers, and it is not used (only defined) in our example. **See also:** Configuring Serial Printers below.

Standard Parallel Operation to device:

```
(echo "$OPENSTR";cat -;echo "$CLOSSTR") >$PTRDEV
```

Standard Parallel Operation to a spooler:

```
(echo "$OPENSTR";cat -;echo "$CLOSSTR") | $PTRDEV
```

Standard Serial Operation to device:

```
(stty $PTRBAUD >$PTRDEV <&1; echo "$OPENSTR";cat -;echo
"$CLOSSTR") > PTRDEV
```

Create a sub-shell to perform the following processes under the process of the script itself:

1. Invoke **echo** to transmit the defined opening string.

2. Invoke **cat** getting its input from standard input (the pipe).

3. Invoke **echo** to transmit the defined closing string.

All of the output from the sub-shell process is optionally piped again through **lptfilter** and finally redirected to the selected device or through the spooler.

If **lptfilter** is required, add the command **| FILTER** immediately following the close parentheses before the **>PTRDEV** or **|PTRDEV** respectively. If not, remove the **| FILTER** command. This increases the speed of the script, preventing an additional process from starting.

By opening the **lpt** printer, we have started the process **sh** (shell) to interpret the script, another sub-shell to perform items 1-3. The sub-shell will have **echo** or **cat** opened and running until the BASIC program closes the channel. Finally, the optional **lptfilter** process may be running. If you have directed output to the spooler, additional processes may also be started.

The entire operation is quite fast, and easily configured. For special applications, you might write in C a printer driver specifically for your needs.

See also: Pipes, lptfilter, filename

Configuring Serial Printers

In the previous section, each time the printer is opened the Unix **stty** command is sent to initialize the device. With some printers, this may cause problems such as overflowing buffers, or losing flow control when the device is turned off-line or out of paper.

If you experience problems with serial printers, check the following conditions:

1. Is the printer set for Xon/Xoff protocol, and if so, does the PTRBAUD definition contain the option for *ixon*?

2. Is the printer set for DTR protocol, and if so, is the wiring correct for the mux, and does the mux support this protocol ?

3. Is the script properly set for serial operation including the Unix **stty** command as the first command within parenthesis?

These conditions should be checked by your installer with a break-out box. You may also have to check with the manufacturer of the printer, system and mux to verify that your configuration and use is supported by the hardware and Unix drivers.

If you continue to have problems:

1. Modify the PTRDEV definition to specify a temporary file for printer output, ie */tmp/printerdata*. Run your report and examine the contents of the file to verify that the data is being correctly sent by the application through the **lpt** script.

2. From *command mode* or shell, use the Unix Commands **stty** and **cat** to configure the port and direct the data to the device:

```
(stty options; cat /tmp/printerdata >/dev/...)
```

3. Once you are able to print data, modify the script using the same parameters remembering to reset PTRDEV to the desired device name.

If printing works, but the printer occasionally loses data or overflows on multiple jobs, it may be necessary to remove the Unix **stty** command from the script. Follow the above example for a parallel printer. Next, add the following code to the system file */etc/rc* or other Unix startup file:

```
(stty ; while : ; do sleep 40000; done ) </dev/...
```

Insert the proper parameters following stty, and </dev/... is the name of the physical device driver, such as /dev/tty23.

It should be noted that these changes are only required on systems redirecting data to a physical device, i.e. PTRDEV, is the actual name of a device driver.

When configuring a printer for use with the spooler, these changes are not required.

Configuring Terminal Drivers

Terminal drivers translate keyboard and display mnemonics between applications and various brands of terminals. When launching a uniBasic process, the value of the environment variable **TERM** selects the terminal translation driver for this session. A filename in the form: *term.name* is opened, where *name* is the value of the **TERM** variable.

Terminal files, typically stored within the *sys* directory, must use a lower-case *filename* and be within a path of the **LUST** environment variable. If a matching terminal driver is not located, an error is printed and no terminal translation functions are available for that session.

Four terminal driver files are supplied for use with your applications; term.ansi, term.tvi925, term.wyse50 and term.wyse60. term.ansi is designed specifically for use with ANSI style terminals and the primary monitor supplied with many systems. The other drivers are for use with Televideo 925, Wyse 50 and Wyse 60 terminals respectively. These may be duplicated and modified for use with other TERM definitions. The Unix cp command may be used to make additional copies of these drivers. For example, to create a Televideo 910 driver, issue the command:

```
cp term.tvi925 term.tvi910.
```

Any standard editor, such as **vi** may be used to adjust the new driver file accordingly.

For a terminal driver to be properly recognized, it must have read-permission enabled and be located within the path specified by the environment variable **LUST**. Once configured, it is recommended that only read-permission remain enabled to prevent corruption.

The names assigned to the **TERM** environment variable are usually defined in the */etc/inittab* or */etc/gettydefs* files. Refer to your Unix system documentation for additional information relating to equating **TERM** names with terminal drivers.

See also: Terminal Translation Files $TERM files

Creating a Customized Installation Media

You may customize the supplied DCI Installation program, **ubinstall**, to include provisions to install your applications, data files, printer and terminal drivers and IQ Data dictionaries.

To ensure proper operation of DCI supplied products, your customized installation procedure should be added to the existing **ubinstall** script. Failure to perform all of the steps contained therein can lead to problems in an installation.

Within the */tmp* directory during installation, the files at the level */tmp/ub* are moved into */usr/bin*, except for the system error message file *errmessage*.

Files at the level */tmp/ub/sys* are moved into **HOME/sys** as defined during installation.

Directories at the level */tmp* are not moved. Directories at the level */tmp/ub* are moved to **HOME/ub**.

Files in *ubdev* (uniBasic Development) are moved under **HOME/ubdev**.

Executable command files in *iq* are moved into */usr/bin*. These include *iq, ddmaint, maketerm, ddconvert,* etc. For a full list of utilities for IQ, please refer to the uniBasic IQ Reference Manual.

Data files required by uniBasic IQ are relocated into the directory **HOME/iq**. A dummy directory **HOME/dd** is created to hold your newly created data dictionaries (as controlled by the environment variable **IQDD**).

A demonstration data dictionary and sample tutorial files are relocated into the directory **HOME/iqdemo** if you chose to install iq demonstration during *ubinstall*.

To create a custom version:

1. Follow the installation instructions on the various DCI supplied installation tapes (omitting the entry of the *ubinstall* command).

2. Move copies of custom printer drivers, system BASIC programs and any other *sys* or LU 0 custom items into */tmp/ub/sys* using the Unix **cp** command.

If you have a complex *.profile*, such as one containing settings which are not prompted during **ubinstall**, place a copy of that *.profile* into */tmp/ub* . It will be necessary to modify the ubinstall script to accommodate this option. Add code in the script following the move of the **errmessage** file to **HOME** to move your custom *.profile* in a similar manner. Be sure that the code is inserted after the initial creation of a **.profile**. Properly coded, installation will replace the default file with your customized *.profile*.

3. Under */tmp/ub*, create any directories that are to be placed under the **HOME** level on your customer's systems. Even if these directories are empty, the **cpio** command will create them for you during installation.

4. Use the Unix **cp** command to move copies of program and data files into the associated installation directories under */tmp/ub*. You may use the **ln** command (link) instead of **cp** to reduce disk space requirements.

5. Use the Unix commands **ls, chown, chgrp, chmod** to verify and set the permissions, *user id*, and *group id* of your directories and files. Verify that your LPT scripts have the *x* attribute (i.e. 555). It is recommended to select a default *group* and *user id*, as is the case with DCI supplied programs and files. During installation, **ubinstall** changes the *group* and *user id* of the supplied DCI files and directories to the prompted owner/manager of the uniBasic installation.

6. Modify the supplied **ubinstall** script to automatically create and/or move your directories to the desired location (optional). Also add code to allow for other directories loaded at the level */tmp* to be installed or moved onto another file system, drive or directory.

7. Your */tmp* directory is now ready to be copied onto a master distribution tape. Issue the following commands from root:

```
cd /tmp
find . -print | cpio -ovc >/device name
```

Note: If you prefer to use the Unix **tar** command, that format is acceptable for your master media. Change your installation instructions accordingly.
