User's Guide to the Primos File System

Perry B. Flinn
Jefferey S. Lee

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

September, 1984

TABLE OF CONTENTS

Foreword

We offer this guide as an attempt to acquaint you with everything you need to know to make effective use of the file system from within the Subsystem. Although we have tried to be thorough in our coverage of concepts and features, we have specifically avoided the details of the programmer's interface to the file system, and everything having to do with implementation. Should you find yourself in need of further information in either of these areas, let us direct your attention to section two of The Software Tools Subsystem Reference Manual, the Reference Guide, File Management System (Prime publication number FDR3110), and the Prime User's Guide (Prime publication number DOC4130).

Introduction

One thing that you will almost certainly encounter frequently during your exploits in the Software Tools Subsystem is the Primos file system. Indeed, there is hardly anything you can do that does not in some way involve this ubiquitous beast.

What is a File?

A file is a named collection of information retained on some storage medium such as a disk pack. Just what kind of information a file contains isn't of much concern to us here; it may be ASCII character codes that form the text of a book or a program's source code, it may be arbitrary binary machine words to be used as input data for a program, or it may be the actual machine instructions of the program itself, to mention just a few. No matter what form the information in a file takes, as far as Primos is concerned it is just an ordered sequence of sixteen bit binary numbers. The interpretation of those numbers is left to other programs.

Entrynames

Since we mentioned that a file has a name, you might ask what names are acceptable. A file is known by something called its "entryname." An entryname is a sequence of 32 or fewer characters chosen from the letters of the alphabet, the decimal digits, and the following special characters:

$ & - * . / _

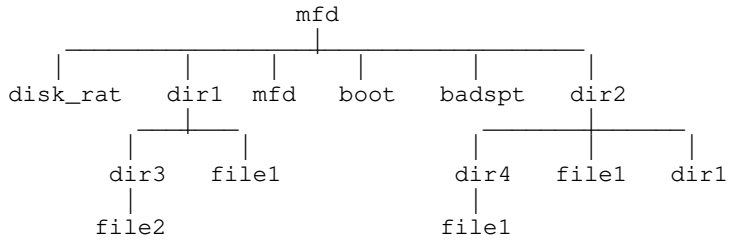
The first character in the entryname must not be a digit. Also, no distinction is made between upper- and lower-case letters. Thus "file_name" and "FILE_NAME" are the same.

Even though Primos allows you to use slashes (/) in entrynames, for reasons that will become apparent in the section on pathnames they must be treated specially when you are using the Subsystem. Because the slash is used to <u>separate</u> entrynames from one another in pathnames, if you want to use it <u>in</u> an entryname you have to "escape" it. By this we mean that you have to precede it with the "escape" character "@". The "@" simply tells the Subsystem to "treat the next character literally, no matter what special meaning it may have;" it is <u>not</u> taken as part of the entryname. It is important that you realize this caveat applies <u>only</u> when you are dealing with the Subsystem; if you try to put an "@" in an entryname when talking directly to Primos, you will get a rather impudent message.

Directories

The way that Primos makes the association between a file's entryname and its contents is through the use of "directories." Like a file, a directory has an entryname and contains some information; but it is different from ordinary files in that the information it contains is treated specially by Primos. The information in a directory is a series of "entries," each consisting of the entryname of some other file, that file's location on the disk pack, and some other stuff that we will cover in a later section. When a file's entryname and location appear in a directory, we say that the directory "contains" that file, or that the file "resides within" that directory. Either way you say it, every file in the system appears in exactly one directory.

Since a directory is so much like a file, there is really nothing to prevent us from having directories that contain other directories. This phenomenon is known as "nesting" and may be carried out to any depth, giving rise to a hierarchical structure:



At the topmost level of the hierarchy is a directory named "mfd", short for master file directory. You will find this directory at the top level of every Primos file system. The MFD is special because it always begins at a fixed location on the disk pack, and because it always contains the following entries:

disk_rat
: The disk_rat (disk record availability table) is a file that catalogs all of the storage space on the disk pack that isn't already in use. It is always the first entry in the MFD and, like the MFD, always begins at a fixed location. This file may have any valid entryname; it doesn't have to be called "disk_rat". But whatever entryname is chosen, it is known as the "packname" for that disk pack.

mfd
: The MFD always has an entry describing itself.

boot
: The "boot" file, which also begins at a fixed location, contains the memory-image of a program that is loaded and executed whenever the computer is cold-started. This program is usually a single-user version of

Primos.

badspt
: Although this file is not necessarily present on every disk pack, if it is it contains a list of faulty records that should not be used.

You may have noticed in the diagram that there are three occurrences of the entryname "file1", and two of "dir1". Each of these entrynames refers to a different file or directory. Even though each entryname must be unique among all those in a given directory, it is perfectly legal to use the same name repeatedly in different directories.

Logical Disks

Since Primos doesn't allow file systems that span multiple disk packs, it does the next best thing and allows you to have multiple file systems in the same installation. Each file system is called a "logical disk" and has exactly the structure described in the last section. Although each installation is virtually guaranteed to have at least one logical disk, the actual number may vary dynamically from 0 to 62. Each disk is uniquely identified by its "logical disk number," and though it is not required, it is extremely desirable for each disk to have a unique packname.

The "Current" and "Home" Directories

Now that we have described this wonderful hierarchy of directories and files just waiting to be used, you might wonder how it is that you go about getting to them. One concept that is central to the solution of this problem is that of the "current directory." From the time you log in to the time you log out, your terminal is having an ongoing relationship with some directory in the file system. When you first log in, this directory is set to whatever the system administrator decided when he created your account. But monogamy is not required; you are free to jump around from directory to directory upon the slightest whim. We say the "current directory" is the directory to which you are attached.

The current directory is important because all the files contained in it are directly accessible to you at the drop of an entryname. In fact, if you are using some of Prime's software, these are the only files accessible to you without changing your current directory. But there is a handy device called the "home directory" that takes some of the edge off of this restriction. Your home directory is the one to which you intend to return after an expedition into the wilds of the file system. In effect, it allows you to remember the location of some particular directory, and to later return there in one giant step, regardless of your (then) current location. Whenever you change your current directory, you get to choose whether to change your home

directory as well or to leave it where it is.

Protection and Access Control

In versions of Primos before Revision 19, to guard your files from unwanted perusal or alteration, the file system included a basic access control mechanism that provided two levels of protection to each file. As part of this mechanism, each directory had associated with it a pair of six-character passwords, one called the "owner password," and the other called the "non-owner password." Normally, when a directory was created its owner password was blank and its non-owner password was zero; these were the default values. But if the passwords had other than default values, then before you could successfully attach to the directory, you had to prove your worthiness to do so by citing one of them. If you cited the owner password, then you were attached to the directory with "owner status;" if it's the non-owner password that you cited, then you were attached with "non-owner status." If you failed to cite either password, then unless one of them had a default value your attempt would be in vain. Just what status you attained when attaching to a directory bears upon the kinds of things you could do to the files it contains.

For the purposes of password protection, there are three things you can do to a file: you can read from it, you can write into it, and you can truncate (shorten) or delete it. Now if you will recall that "other stuff" we mentioned a while back as being in a file's directory entry, part of it is two sets of "protection keys:" one for people attached to the containing directory with owner status, and the other for those with non-owner status. Each set of keys has a bit for each type of access: read, write and delete. If a bit is turned on, the associated type of access is permitted; otherwise, it is denied.

Revision 19 of Primos introduced Access Control Lists (ACL's). Unlike the password protection previously described, ACL's allow specific permissions on files to be granted on a per-user basis, instead of a broad class of permissions being granted to anyone who happens to know, or guess, the password. They also allow better control over permissions given to users. Previously, in order to allow a user to create files in a directory, he was implicitly given the right to delete any other files in that directory, also. With ACL's, this is no longer the case.

An ACL consists of a list of up to 32 identifiers and privileges associated with each of the identifiers. An identifier can be a user's login name or it can be a group identifier associated with several users. If a user's name and associated group are both in an ACL, the user's login name takes precedence. The seven different privileges associated with ACL's are:

add This privilege is associated with a directory and allows the user to create a new file within that

directory. Once the file is created, the user has full read/write access to the file until the file is closed, at which point other privileges determine the accessibility of the file.

delete This privilege is associated with a directory and allows the user to delete an existing file within that directory.

list This privilege is associated with a directory and allows the user to list the contents of the directory (like with 'lf').

protect This privilege is associated with a directory and allows the user to set ACL protection for objects in the directory.

read This privilege is associated with a file and allows the user to open a file for reading or to execute a file. The user must first be able to attach to the directory before he can read the file, which implies use privilege (see below).

use This privilege is associated with a directory and allows the user to attach to the directory (like with 'cd'). In order to access a file or a directory, the user must have use privilege on all intervening directories between the MFD and the desired file or directory.

write This privilege is associated with a file and allows the user to open a file in write mode or to truncate a file.

Associated with the ACL is its type. There are five different types of ACL's. The first type is the specific ACL. This gives protection on one specific file object and is associated with only that object. If the object is deleted then the specific ACL goes away, also.

The second type of ACL is the default specific ACL where a specific ACL is set on an ancestor directory of the current object. If the object is not protected by a specific ACL or an access category (the next type), then it is given the same protection as the ancestor directory.

The third type of ACL is the access category ("acat"). An access category, unlike the two previous types, may protect many objects at one time with the same protections. An acat appears in the file system as a file that cannot be read or written, and its name must end in ".acat". It is a separate type of file system object (just as in 'lf -l' listings, DAM files are different from SAM files -- acats are of type ACT). An access category need not protect any object since it exists independant of any other object in the file system. If an access category is deleted, any object that it was protecting becomes default

protected, or becomes protected by the directory that contains it.

The fourth type of ACL is the default access category. This is an access category that protects a directory that contains other objects that are then protected by default.

The last type of ACL is the priority ACL. This is an ACL that is set on an entire disk partition by the system administrator, normally at boot time. Any rights given by a priority ACL override any rights given by any other ACL's.

In order to allow for a gradual change from the older versions of Primos to Revision 19, it is possible for password directories and ACL's to exist in the same system, although password directories will eventually be unsupported. There is a restriction in that ACL directories may contain both password and ACL directories but password directories may not contain ACL directories. In order for any directory to be an ACL directory on a logical disk, the MFD of that partition has to be ACL protected. Password directories also overcome some of the limitations of ACL's. If an ACL gives someone the privilege of writing a file, then under all circumstances they are allowed to write the file. If the file is in a password directory, though, they may only write the file if they know the password. This means that a password can be nested deep in a program that is used to control their access to a file, even if the person running the program does not know the password.

Pathnames

Unlike the Prime software we mentioned that only lets you manipulate files in your current directory, the Subsystem places no restrictions on the whereabouts of the files you can reference. Generally speaking, anywhere the name of a file is required you may use something called a "pathname." A pathname is a construct that allows you to uniquely specify any file in the system by describing a path to it from some known point. As we have seen, the current directory is one such point, and because of its fixed location, the MFD on each logical disk is another.

The syntax of a pathname is divided into two basic parts which we will call the "starting node," designating the particular known point at which the path starts, and the "directory path," designating the actual series of nested directories that leads to the desired file. Both parts, by the way, are optional: either one may stand alone, they may stand together, or they may both be omitted. But if both are present, they must be separated by a single slash (/).

The starting node of a pathname comes in two varieties. The first designates the MFD of a particular logical disk and consists of an initial slash followed by a packname, a logical disk number in octal, or a single asterisk (*):

```
/vol00
/7
/*
```

If the asterisk is used, the MFD of the logical disk containing the current directory is implied; the other two forms should be self-explanatory. The second variety of starting node refers to one of the current directory's ancestors in the hierarchy and consists of one or more backslashes (\). The number of backslashes indicates the number of nesting levels above the current directory at which the path begins. If the starting node is omitted altogether, then the path starts in the current directory.

Now the other half of a pathname, the directory path, is simply a series of one or more entrynames, each separated from the next by a single slash. The first entryname must be contained in the starting directory, and each subsequent entryname must reside in the directory designated by the preceding entryname. The very last entryname in the path is that of the target file. To illustrate,

```
src/lib/swt
extra
```

are proper directory paths. As you might expect, if the directory path is omitted, the target of the pathname is the starting directory. Thus, the pathname from which both the starting node and the directory path have been omitted (the empty pathname) refers to the current directory.

A couple of special cases are worth mentioning here: First, a pathname that begins with a slash and whose directory path is not omitted need not contain a packname or logical disk number. In this case an implicit search of the MFD on each logical disk is made for the first entryname in the directory path. The MFD on the lowest numbered logical disk in which that entryname is found is taken as the starting directory. Notice that such a pathname is easily recognizable because it begins with two slashes; the first one belongs to the starting node and the second separates it from the directory path:

```
//system
```

The second special case has to do with pathnames beginning with a backslash. Although we said that a slash must be used to separate a starting node from a directory path, when using backslashes the intervening slash is not required; indeed it is omitted more often than not.

Passwords in Pathnames

The following discussion is applicable only for password protected directories, since ACL protected items do not need pas-

swords. Thus far in discussing pathnames we have assumed that we may freely specify any valid sequence of directories in a directory path without regard to the passwords that may be associated with those directories. In fact, this is true only if the directories have at least one password with a default value, or if the directories are ACL directories. You see, the interpretation of a pathname involves temporarily attaching to each directory in the path; if this can't be done without a password then the pathname can't be interpreted. Furthermore, the set of access privileges (owner or non-owner) available to you with respect to the target file is determined by whether you are attached to its parent directory as an owner or a non-owner by the pathname interpreter. So, to let you deal effectively with passworded directories, the pathname syntax allows you to append a password to each directory entryname in the path, separated from the entryname by a colon:

```
entryname:passwd
```

If a password is so specified, the pathname interpreter will use it when attaching to the associated directory.

A password may contain arbitrary characters which are not necessarily legal in entrynames. So to avoid the ambiguity in interpreting a password containing a slash, as with entrynames, the slash must be "escaped" by preceding it with an "@". This also means that the "@" itself must be escaped if it is to appear literally in the password. Remember that the "@" used as an escape character is not included in the password; it merely turns off the special meaning of the character that follows.

The following set of examples contains an instance of just about every possible variation in the syntax of pathnames, along with an explanation of each. A formal summary of pathname syntax in BNF notation is included in Appendix B.

a_file
: A file in the current directory whose entryname is "a_file".

a_ufd/a_file
: A file whose entryname is also "a_file" and is contained in the subdirectory "a_ufd" of the current directory.

\
: The parent of the current directory.

\brother (or \/brother)
: The file or directory named "brother" that resides in the same directory that contains the current one.

/0/cmdnc0:secret
: The directory named "cmdnc0" (one of whose passwords is "secret") which resides in the MFD on logical disk 0.

/md
The MFD on the logical disk whose packname is "md".

/*/boot
The "boot" file on the current logical disk.

//spoolq/q.ctrl
The file named "q.ctrl" in the "spoolq" directory on the lowest numbered logical disk that has one.

ki@/da:ad@/ik
The directory residing in the current directory whose entryname is "ki/da" and one of whose passwords is "ad/ik". (Note the use of the "@" to turn off the special meaning of "/".)

<empty>
The current directory.

Templates

In order to provide flexibility in the organization and placement of the directories and files used by the Subsystem, the pathname interpreter contains a primitive macro substitution facility, a feature that is loosely referred to as "templates." Templates provide a means for designating particular files or directories without having to know their exact location in the file system, and for constructing file names whose exact interpretation may vary with the context in which, or the user by whom they are used. A template is constructed from letters, digits and underscores and is always enclosed in equals bars (=). (Templates do not have to begin with a letter). Unlike entrynames, upper- and lower-case letters are different in template names; "name" and "NAME" are not the same. Each defined template has an associated value which is an arbitrary character string. The effect of including a template in a pathname is the same as if its value had appeared instead.

There are two types of templates: static and dynamic. The value of a dynamic template varies depending upon who you are, how you are connected to the computer, or what time it is. The following list describes all of the available dynamic templates:

=date=
The current date in the format mmddyy.

=day=
The current day of the week; "monday", for example.

=home=
The current user's initial login directory (set by the system administrator when he created the account). This may vary on a per-user per-project basis. I.e., the system administrator may set it up so that the initial login directory for a given user is different

for different projects.

=passwd=
The owner password of the current user's profile directory. (This is the same password the Subsystem asked you for when you typed "swt".)

=pid=
The current user's process-id. This is a three-digit number in the range 001-128 that is unique to each logged-in user.

=time=
The current time in the format hhmmss.

=user=
The current user's login name.

These templates are particularly useful for constructing unique file names.

Static templates are those whose definitions are independent of the context in which they are used. These templates and their values come from two sources. The file whose name is the value of the template

=template=

contains system template definitions that apply globally to all Subsystem users. In fact the definition of "=template=" itself is contained in this file, as are definitions for other important Subsystem files and directories. In addition to this file, you may have in your profile directory (named by the template "=varsdir=") a file named ".template" that contains your own personal template definitions. Any templates that you define yourself preempt similarly named system templates, so you should exercise caution in choosing names. Also note that any new templates you place in your personal template file do not take effect until the next time you enter the Subsystem via 'swt'; this is the only time that the file is examined. If you wish to create templates that will take effect immediately, use the 'template' command (do a 'help template' for details).

The format of both files is the same: a series of lines containing a name, followed by one or more blanks, and then a value. Blank lines are ignored, as are leading and trailing blanks on each line. Comments may be introduced with the sharp character (#); all characters from the sharp to the end of the line are ignored:

```
# example of a template definition
   macros    //smith/misc/macros     #Smith's macros
```

The above example defines a template "macros" referring to the file "//smith/misc/macros." A quick perusal of the contents of "=template=" should clear up any lingering questions you may

have. Just for convenience, all dynamic and system templates, along with an explanation of each, are listed in Appendix A.

If you look at the template definition file, you will notice that some of the definitions appear to contain templates themselves. This is perfectly legal, for after each template is expanded, the result is inspected for further templates until no others are found. This makes possible the definition of such templates as "=varsdir=", and generally enhances the utility of the mechanism.

Just one further remark about templates: Remember the trouble we had with "/" in passwords and entrynames? Well, we have a similar situation with "="; when should it be taken literally, and when should it indicate the beginning of a template? To solve this dilemma, any time the template expander sees a template with an empty name (that is, two consecutive equals bars), it supplies a single "=" as the replacement value and does not consider it to be the start of another template. So if you ever want a literal "=", in a password for example, just type "==" and you've got it.

Device Names

Up to this point, we have been talking only about disk files, and the pathnames we have described have corresponded exactly to some actual sequence of directories leading to a file. Although this is certainly the most common use of pathnames, there is one additional feature that significantly enhances their usefulness. If the "starting node" of a pathname is "/dev", the pathname doesn't necessarily refer to a disk file, but may instead refer to an arbitrary peripheral device, or to some special file that requires unusual processing. As with ordinary pathnames, the "directory path" provides more information about the target file or device.

Perhaps the most useful of these extended pathnames (or "device names," as they are usually called) is

```
/dev/lps
```

which refers to the line printer spooler. When this pathname is opened for writing, a special disk file is created and other processing is done so that when the file is closed, its contents will be written to the on-site line printer by the spooler and then deleted. Additional entrynames may be included after the "lps" to select various processing options specific to the spooling process. A complete list of these in included as Appendix C.

Another useful device name is

```
/dev/tty
```

which refers to your terminal device. There are also others which, when opened, yield file descriptors for the various stan-

dard input and output ports:

/dev/stdout	/dev/stdin
/dev/stdout1	/dev/stdin1
/dev/stdout2	/dev/stdin2
/dev/stdout3	/dev/stdin3
/dev/errout	/dev/errin

Finally, the device name

/dev/null

when opened yields a file descriptor which discards all data written to it and returns an end-of-file signal every time it is read. It is really just a fancy name for the proverbial bit bucket.

Georgia Tech Extensions

As many of you reading this guide will eventually come to know, using the standard Primos file system can be quite awkward, principally because of the constant necessity of typing passwords in pathnames. Relief from this burden comes only at the expense of security, which in many cases is a more important consideration than ease of use. So that we can have our cake and eat it too, we at Georgia Tech have made a few modifications to the standard protection mechanism that virtually eliminate the necessity for typing passwords in all but the rarest of circumstances. The Subsystem requires none of these modifications to operate properly, and in those cases where it behaves differently depending on the extant version of Primos, it does so completely transparently to the user.

In Georgia Tech Primos, if a directory's owner password is a valid entryname, it is assumed to be the login name of the user that "owns" that directory. In this case, the "owner password" is instead called the "owner name." When you attach to a directory whose owner name "matches" your login name, you automatically get owner status without having to cite a password. This is the only difference between the protection mechanism in Georgia Tech Primos and the standard mechanism. In all other situations, you can expect the standard behavior.

Appendix A - Standard Templates

The following list describes all of the templates that are provided either in the standard Subsystem template file or by the template interpreter.

=aux=
: This Subsystem directory contains large files that are not absolutely necessary for the operation of the Subsystem.

=bin=
: The standard Subsystem command directory.

=bug=
: The directory in which the Subsystem bug reporting mechanism collects bug reports.

=cldata=
: Defines the location of the Primos CLDATA structure, used internally by the Subsystem command interpreter (shell).

=cmdnc0=
: The directory to which the system console is normally attached.

=crondir=
: The directory where the 'cron' program creates temporary files for phantoms.

=cronfile=
: The file that contains the directive lines for the 'cron' program.

=date=
: The current date in the format mmddyy.

=day=
: The current day of the week (e.g., "monday", "tuesday", etc.).

=dictionary=
: A file containing English words, used by the spelling checker.

=doc=
: The Subsystem documentation directory.

=ebin=
: A directory of programs called by shell programs in "=bin=".

=extra=
: A standard Subsystem directory containing miscellaneous files required for proper operation of the Subsystem.

=fmac=
The Subsystem directory containing all the text formatter macro definition files.

=GaTech=
This is a template having nothing to do with pathnames. Its value is "yes" at installations that run the Georgia Tech version of Primos, and "no" elsewhere. Programs that are sensitive to the operating system version use this template to determine their environment.

=gossip=
The directory containing user-to-user message files generated by the 'to' command.

=histfile=
The current user's saved command history file.

=home=
The current user's login directory. Take note that this is not the same as his "home directory" as described in the section on "current" and "home" directories.

=incl=
The standard Subsystem directory containing files that are included by Ratfor and C programs.

=installation=
A file containing the name of the installation.

=lbin=
The standard Subsystem locally-supported command directory.

=lib=
The Primos directory containing all library files that should be accessible to the loader.

=mail=
The Subsystem directory that contains per-user mail delivery files.

=mailfile=
The current user's mail storage file. This is where the 'mail' command deposits a letter after you have asked that it be saved.

=new_words=
If this template exists and describes a legal file name, the 'spell' program will write a copy of unrecognized words to this file.

=newbin=
The Subsystem directory into which newly-compiled commands are placed during a recompilation of the entire Subsystem.

=newcmdnc0=
The Subsystem directory into which newly-compiled Subsystem files that belong in "cmdnc0" are placed during a recompilation of the entire Subsystem.

=newebin=
The Subsystem directory into which newly-compiled commands destined for "=ebin=" are placed during a recompilation of the entire Subsystem.

=newlbin=
The Subsystem directory into which newly-compiled locally-supported-commands are placed during a recompilation of the entire Subsystem.

=newlib=
The Subsystem directory into which newly-compiled object code libraries are placed during a recompilation of the entire Subsystem.

=news=
The directory used by the Subsystem news service.

=newsfile=
The current user's news delivery file.

=newsystem=
The Subsystem directory into which newly-compiled Subsystem files that belong in "system" are placed during a recompilation of the entire Subsystem.

=passwd=
The password of the current user's profile directory. (This is the same password the Subsystem asked you for when you typed "swt".)

=pid=
The current user's process-id. This is a three-digit number in the range 001-128 that is unique to each logged-in user.

=src=
The Subsystem source code directory.

=srcloc=
A file associating each Subsystem library subroutine and command with the pathname(s) of its source code file(s).

=statistics=
The system template which controls whether or not command statistics are to be kept. (See the "Application Notes" section of the Command Interpreter User's Guide.)

=statsdir=
The directory where command statistics are recorded. (See the "Application Notes" section of the Command Interpreter User's Guide.)

=syscom=
The directory where the Primos subprogram keys (predefined constants) are stored.

=sysname=
This is the system's Primenet node name, if it is a network system.

=system=
The Primos directory that contains the core-images of the various shared memory segments.

=temp=
The Subsystem directory in which all temporary files are created.

=template=
The system template definition file.

=termlist=
A file describing the location and type of each terminal connected to the computer.

=time=
The current time in the format hhmmss.

=ttypes=
A file containing a list of terminals supported by your Subsystem and their characteristics.

=ubin=
By convention, the user's private command directory.

=user=
The current user's login name.

=userlist=
A file containing a list of all users authorized to use the computer.

=utemplate=
The current user's private template definition file.

=vars=
The Subsystem directory in which all per-user profile directories are contained.

=varsdir=
The current user's profile directory.

=varsfile=
The current user's shell variable storage file.

=vth=
The directory used by the Subsystem virtual terminal handler.

Appendix B - Pathname Syntax

For the grammar aficionados among you, here is a formal description of the syntax of pathnames. The notation used is an extended Backus-Naur Form (BNF) which is described in the introduction to the Software Tools Subsystem Reference Manual.

```
<pathname>       ::= <starting node>
                   | <directory path>
                   | <starting node>/<directory path>
                   | <empty>
<starting node>  ::= \{\}
                   | /<volume id>
<volume id>      ::= <packname>
                   | <octal integer>
                   | *
<packname>       ::= <entryname>
<directory path> ::= <node>{/<node>}
<node>           ::= <entryname>[:<password>]
<entryname>      ::= <non-digit>{<valid char>}
<non-digit>      ::= <letter>  |  <special char>
<valid char>     ::= <non-digit>  |  <digit>
<letter>         ::= a | b | c |...| x | y | z
<digit>          ::= 0 | 1 | 2 |...| 7 | 8 | 9
<special char>   ::= # | $ | & | - | * | . | / | _
```

Appendix C - Spool Options

The entrynames that may be appended to the "/dev/lps" device name to control spooling options are summarized in the following list. These entrynames correspond exactly to the options that are accepted by the 'sp' command (see section one of the Subsystem reference manual). These entrynames and associated values must be separated by slashes or blanks, e.g. "/dev/lps/b/TECH/" or "/dev/lps/b TECH."

a This option selects a specific location at which the file is to be printed. The immediately following entryname in the path is taken as the name of the destination printer.

b The file name that is printed on the banner page of the printout may be set arbitrarily with this option. The next entryname in the path is taken as the name to be printed. If this option is not used, the name "/dev/lps" is printed.

c This option specifies the number of copies of the file that are to be printed. The next entryname must be a decimal integer indicating the number of copies.

d Printing of the file may be deferred until a specific time of day using this option. The next entryname in the path must be a time of day in any reasonable format.

f If specified, this option indicates that the print file contains standard Fortran carriage control characters.

h This option causes the spooler to suppress the printing of the banner page that normally precedes each printout.

j Specifying this option causes the spooler to suppress the trailing page eject that it normally supplies at the end of each printout.

n This option causes the spooler to print a consecutive line number in front of each line of the print file.

p This option instructs the spooler that the print file is to be printed on a special type of paper. The name of the desired form should follow as the next entryname in the path.

r "Raw" forms control mode is selected by this option. No carriage control characters are recognized, nor is any pagination done when this mode is in effect.

s This option selects the standard Primos forms control mode. Under this mode, the printout is automatically paginated, and a header line is printed on each page.