I.2-UNIX-APPSDFLT DESCRIPTION OF THE APPS DEFAULTS SYSTEM

Purpose

When application programs and scripts are written, values for variables need to be assigned. The potential exists for these values to change at some point in the future. To avoid rewriting programs and scripts when these values need to be reset, a general use system was developed to centralize the assignment and the maintenance of these values. This system is called 'Apps_Defaults' (AD). Many of the scripts and programs that run on AWIPS use this system to set execution controls and path names. AD has the added benefit of easily overriding system default values for development or testing purposes.

The AD system is available for use from the command line in UNIX and as a function either in FORTRAN or C. Examples of both methods are provided at the end of this document.

Whether the AD system is used from the command line or as a function call, the concept is the same:

> Make a <request> for a variable value (within AD the variable is also known as a token) and receive a <reply>.

The process for getting the value (<reply>) for the requested token (<request>) is as follows:

1. If an environment variable matching <request> is defined then <reply> is the value of the environment variable.
2. If <request> is found as a match in a file that contains token-resource (t-r) relationships then the value is the resource value. Files defined by the following environment variables are scanned in the following order to find the first token that matches <request>:
   APPS_DEFAULTS_USER ..... a user's set of tokens
   APPS_DEFAULTS_PROG ..... a program specific set of tokens
   APPS_DEFAULTS_SITE ..... a site wide set of tokens
   APPS_DEFAULTS ..... a system wide set of tokens
   On AWIPS, the first, third and fourth environment variables are set by the login process. Note that the first will be set only if an AD file is found in the user's home directory. On AWIPS, the values for the third and fourth files are:
   APPS_DEFAULTS_SITE=/awips/hydroapps/.Apps_defaults_site
   APPS_DEFAULTS=/awips/hydroapps/.Apps_defaults
   The second file in the sequence is used primarily for development and testing purposes and is not defined by default on AWIPS systems.
3. If <request> is not found then <reply> is set to a null string.

AD File Syntax

Each file is read checking for the first match between <request> and

a defined token. The file syntax is:

<token> <delimiter> <resource>

where

<token>	is a string delimited by white space (blanks, tabs, etc) or <delimiter>
<delimiter>	is a : (colon)
<resource>	is any string - the value returned depends on the following conventions:

(1) a valid token-resource requires a valid token followed by a valid resource
(2) the token-resource relationship must be contained on a single line
(3) no white space needs to surround <delimiter>,
(4) comments are indicated by a #
(5) <resource> can contain white space or a # if it is enclosed by ' or "
(6) blank lines are allowed in the file
(7) refer-backs are indicated by $(...) - the '...' is set the same way as any other token and is substituted for the $(...) string to set the final resource value
(8) multiple refer-backs are allowed in <resource> but embedded (i.e. $($(...)) refer-backs are not allowed
(9) the first value of <token> matching <request> is returned even if null

The following is a sample AD file:

```
#---------------------------------------------------------------------
# This, the previous, and the following lines are comment lines.
# Blank lines are valid.

ofs_level     : testcase         # this is a comment on valid t-r
ofs_reor_lvl  : test:reor        # ':' allowed in body of <resource>
ofs_inpt_grp  : "test  case"     # white space allowed in <resource>

ofs_file_grp  : /home/$(ofs_level)/files # refer-back to prior token -
                                         # returned resource will be
                                         # /home/testcase/files

ofs_xxx       xxx       # invalid - no delimiter - would be ignored
ofs_yyy    : #yyy       # invalid - no resource  - would result in null
ofs_yyy    : "#yyy"     # valid

# This and the next line are comment lines.
#---------------------------------------------------------------------
```

Command Line Use Of The AD System

Program **get_apps_defaults** (or **gad**) can be used to print the value of an AD token. The syntax is:

get_apps_defaults <token>
or

```
gad <token>
```

The script **gad_w** can be used to list the resource levels and if found the token-value pairs and the actual value in the AD file. The syntax is:

```
gad_w <token>
```

Note that **get_apps_defaults**, **gad**, and **gad_w** all require the entire token, and will only provide a response if a perfect match is found. These commands are provided with AWIPS and should be in the user's execution search path.

The script **go** can be used to go to the directory for the specified AD tokens. For example:

```
go ofs_rls
```

The script **gohelp** can be used to list the value of the environment variables and tokens defined in the AD files. The syntax is:

```
gohelp <string>
```

If <string> is not specified, all environment variables and tokens are listed. If <string> is specified, the token-value pairs are listed only for those environment variables or tokens that contain <string>.

Note that the **go** and **gohelp** commands will work with the Korn or POSIX shells and not in the C shell. They are defined as shell functions which are not available in the C shell. These inline functions get defined by the execution of another script, **fun_go**, which must get run when the user logs on to the system.

Note also that the **go** command will work with only a portion of the token, taking you to the directory indicated by the first response to resolving the token. The **go** command will fail if the token reply is not a valid directory.

Use Of The get_apps_defaults Function

The function **get_apps_defaults** can be used to get the resource value for a token.

The calling sequence in FORTRAN is:

```
CALL GET_APPS_DEFAULTS (TOKEN,LTOKEN,REPLY,LREPLY)
```

```
where TOKEN   is a CHARACTER*(*) input variable that contains the
              token variable name
      LTOKEN  is an INTEGER*4 input variable that contains number
              of characters in TOKEN
      REPLY   is a CHARACTER*(*) output variable that contains
              value of TOKEN
```

LREPLY is an INTEGER*4 output variable that contains number of characters in REPLY

The prototype in C is:

```
int get_apps_defaults (char *TOKEN, int *LTOKEN, char *REPLY,
                       int *LREPLY)
```

where the returned function value is:
0 = REPLY is not null
1 = REPLY is null