MILTIE: PL/I and SNOBOL4 Interfaces
for Terminal I/O through MILTEN

Introduction

Presently the dial-up IBM 2741 terminals connected to the 360/91 are managed by a supervisory program MILTEN. WYLUR is an example of a subsystem which utilizes MILTEN for all terminal input/output. In order to facilitate running jobs which can read or write to a terminal, a set of assembly language routines have been written which handle the task of communicating with MILTEN. These two programs (both called MILTIE) have most of their code in common but differ in order to handle properly the differences in parameter passage and subroutine return mechanism between SNOBOL4 and PL/I. The original MILTIE satisfied the SNOBOL4 specification of having only one entry point per module. This restriction has been reflected in the PL/I version for compatibility.

Facilities provided

MILTIE will permit communications to one terminal at a time. This terminal need not remain fixed, nor are there constraints on the user name, etc. allowed to signon onto a subsystem. A user subsystem may have MILTEN require a password ("magic word") prior to allowing a user to signon a subsystem. This word may be changed (or computed) dynamically if desired. In addition, a subsystem may learn the identity of the signed-on user from MILTEN and have "intruders" transferred back to WYLUR or MILTEN. This will permit further signons. If a user is accepted by MILTIE, new signons (even if the password barrier is passed) will be rejected, and the terminal will be reconnected to WYLUR.

There are five major services provided by MILTIE.

Initialize and Set Magic Word informs MILTEN that a subsystem exists.
(The subsystem name is the jobname from the job card.)

Transfer Control will cause the signed-on line to be transferred to a different subsystem (e.g. WYLUR). At this point MILTIE will permit another terminal to sign on.

Uplevel is used to cause the execution of a MILTEN command. This may be done to utilize the services of MILTEN, but a common usage is for a subsystem to forward all unrecognized commands to MILTEN. If it isn't a valid MILTEN command either, the subsystem will be so informed.
Write causes a string of text to be typed at a terminal. The subsystem must supply a carriage-return character (CR - X'15') if he desires this action, for example. On the other hand both the use of tab stops and lower-to-upper case conversion will be handled by MILTRAN (if desired).

Converse is the most sophisticated of the major functions. You may specify two text strings for output (write-text and prompt-text) and the characters typed by the user will be returned to the subsystem. (Tabs will be replaced by the appropriate number of blanks, lower-case made upper-case, if specified, and in all cases the final carriage-return will not be included in the read-text.

Converse acts as follows: the write-text is transmitted. Explicit use of CR is necessary. As we will see later, we may or may not allow the use of Attention (ATTN) to terminate the typing of this information. If not permitted, when ATTN is hit, a CR is sent out, and the write-text is written again. If ATTN is permitted and is hit, a CR is supplied and the prompt-text will be typed. The keyboard will now unlock, enabling a user to input information. Backspace will "erase" all characters backed over. Tabs, lower case, etc. may all be used if the system is properly initialized. If a user hits ATTN during prompt-text or while typing on the keyboard, the subsystem will be able to determine that fact, and the text entered until the use of ATTN.

There are four mode switches which may be set by the subsystem. These determine:

1. should lower-case be converted to upper-case automatically (on input)?
2. should the user be allowed to hit ATTN during write-text?
3. should messages from other users be allowed to interrupt and type out?
4. should tab stops be used when typing out on the terminal?

(Note: Tabs may be set or reset only in WYLIEIR at present, but may be used later by other subsystems.)

There are five miscellaneous information strings available concerning the identity of a signed-on user. They are:

1. user account number (ex: 'CG')
2. user initials (ex: 'HJS')
3. terminal ID (ex: 'S03')
4. whether terminal lacks 'reverse break' feature ('1' is yes, no feature, '0' is no, feature present)
5. does user wish to keep 'upper-case backspaces' as 'backspaces'? ('1' is yes, '0' is no)
A special call is provided for determining the nature of a failure condition (e.g. ATT\(\star\) was hit, terminal I/O error, bad parameters, etc.)

**Actual Calling Sequences**

*SNOBOL\(^4\) calling sequences are in the form:

\[
\text{STRING} = \text{MILTIE(STRING,STRING,STRING)}
\]

As in all SNOBOL\(^4\) function calls omitted arguments will have null strings supplied as values.

The PL/I version requires calls of the form:

\[
\text{CALL MILTIE (STRING,CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, FIXED BIN(15))};
\]

**Converse:**

\[
\begin{align*}
\text{(SNOBOL}\(^4\))\text{ read-text} &= \text{MILTIE (1, write-text, prompt-text)} \\
\text{(PL/I) CALL MILTIE (1, write-text, prompt-text, read-text, error-code)};
\end{align*}
\]

Converse operates as described previously. The prompt-text should be \(\leq 32\) characters. Write-text must be \(\leq 133\) characters.

Errors in processing converse will cause a **FAILURE** return in SNOBOL\(^4\). The exact cause may be ascertained (as we will see) using MILTIE (35). The PL/I package will set error-code = 0 for no errors, and to the cause of error number (see MILTIE(35) for a list) otherwise. This method of signalling errors is used in all calls to MILTIE and will not be mentioned for other functions.

**Write:**

\[
\begin{align*}
\text{(SNOBOL}\(^4\))\text{ MILTIE(2,write-text)} \\
\text{(PL/I) CALL MILTIE (2,write-text, dummy, dummy, error-code)};
\end{align*}
\]

Write operates as described previously. Write-text must be \(\leq 133\) characters.

**Up-level:**

\[
\begin{align*}
\text{(SNOBOL}\(^4\))\text{ user-id} &= \text{MILTIE (6, command)} \\
\text{(PL/I) CALL MILTIE (6, command, dummy, user-id, error-code)};
\end{align*}
\]

Up-level will pass a command to MILTIE\(\mathrm{\text{\textsc{-}}\text{E}}\) for processing. A null command will cause MILTIE\(\mathrm{\text{\textsc{-}}\text{E}}\) to prompt SYSTEM? at the terminal. When MILTIE receives an up-level order it temporarily releases the current user from the subsystem. Normally, once the command is executed (or is discovered to be invalid), the user is automatically signed-on again to the subsystem.
It is possible that a user may interrupt processing during the up-level, or another user may somehow pass the password at this moment. In either case, the next user signing on (automatically or not) will be accepted and his user-id returned to the driver program. Other users will then be transferred to WILBUR (until this user has been released).

An illegal MILTIE command will cause FAILURE (or set the error-code).

Transfer Control

(SNOBOL4) MILTIE (7, system-name)

(PL/I) CALL MILTIE (7, system-name, dummy, dummy, error-code);

System-name must be \leq 8 characters; otherwise MILTIE signals an error in the appropriate manner. If the system is not known to MILTEN it will prompt SYSTEM? but the subsystem will not learn of this. After transferring control, an "Initialize" call will accept a new signon.

Initialize and Set Magic Word

(SNOBOL4) user-id = MILTIE (13, magic-word)

(PL/I) CALL MILTIE (13, magic-word, dummy, user-id, error-code)

Several functions are wrapped into one in this call. The first time it is used, it announces the presence of a subsystem to MILTEN. At this point the MILTEN command SHOW SYSTEMS will type the jobname (among others). To enter a subsystem (from WILBUR or MILTEN) simply type the jobname (i.e. subsystem name). The magic-word is a password of 1 to 8 characters. More than eight will signal failure by MILTIE. A null string as magic-word will cause no prompting for a password when a user attempts to signon.

Once "Initialize" is issued the subsystem sleeps until a user signs-on. At that time it is awakened, and the three letter user-id is returned to the subsystem for inspection.

"Initialize" may be used more than once by a subsystem. (It does not truly re-initialize the subsystem however). If a line is signed-on when "Initialize" is called, it is transferred to WILBUR at once. The magic-word specified replaces the present magic-word. (It may be the same or changed). If the previous line has been transferred away, "Initialize" will set a magic-word and await a sign-on.

Calls for User Information

(SNOBOL4) info = MILTIE (nn)

(PL/I) CALL MILTIE (nn, dummy, dummy, info, error-code);

Five calls provide various details on the user and his terminal.
nn info
21 4 character account number (padded with blanks on the right)
22 3 character user initials
23 3 character terminal id number
24 '1' if no reverse break allowed, '0' if allowed
25 '1' if upper-case backspace kept, '0' if treated as "erase"

These five calls should always be successful.

Modifying Terminal Behaviour

Four calls permit the user to set switches which affect terminal behaviour (such as lower-case to upper-case conversions). Some of these may be changed using the "up-level" command but that is a slower means of accomplishing the modifications. Passing a string of length zero (null string) will force the switch off and any non-null string sets the switch on.

(SNOBOL4) MILTEE (nn, on-off-string).

(PL/I) CALL MILTEE (nn, on-off-string, dummy, dummy, error-code);

nn = 31: When on, lower-case alphabetic input is converted to upper-case during future Converses. When off, all text is passed to the subsystem as typed. This switch is initialized to ON (do conversions) whenever a new user signs-on to the subsystem. It will not be reinitialized after an up-level (unless a different user happens to sign-on at that time). SET UPFEH/UFLOW are the equivalent MILLEN commands for this operation.

nn = 32: When on, write-text from future Write or Converse operations will not be terminated by the user of ATTN. If off, a user may hit ATTN to terminate typing. This switch is initialized to OFF (permit ATTN). There is no MILLEN equivalent for this function.

nn = 33: When this switch is on, no messages (except urgent operator messages) will interrupt the typing at a terminal. This function is performed during a WYLBUR "CLEAN" listing although there is no mode switch available in MILLEN or WYLBUR which affects more than one listing. If this switch is off, messages may interrupt terminal I/O. The switch is initialized to OFF (permit messages).
nn $= 3^4$: When this switch is ON, any physical tab settings will not be used on output. This is identical to the command SET SLOWLIST. If this switch is OFF, tabs will be used wherever possible on output (SET FASTLIST). This switch is initialized to OFF (use tabs on output). Note that tabs may be set and made known to MILTEN only through the use of appropriate WILBUR commands.

These four operations have no failure conditions.

Determining why an operation failed

(SNOBOL4) reason = MILTIE (35)

(PL/I) CALL MILTIE (35, dummy, dummy, reason, error-code);

After an unsuccessful operation, you may use this call to MILTIE to ascertain the cause of failure. This must be the next call to MILTIE, or this information will be lost. Reason is returned as a string of integer characters. The indicators are:

<table>
<thead>
<tr>
<th>reason</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ATTN hit during prompt</td>
</tr>
<tr>
<td>2</td>
<td>ATTN hit during read</td>
</tr>
<tr>
<td>3</td>
<td>Terminal I/O error</td>
</tr>
<tr>
<td>4</td>
<td>Use of unset tab on input</td>
</tr>
<tr>
<td>5</td>
<td>The operation was interrupted by a break (use or operator message)</td>
</tr>
<tr>
<td>6</td>
<td>Phone was hung up</td>
</tr>
<tr>
<td>7</td>
<td>Logoff</td>
</tr>
<tr>
<td>8</td>
<td>Die</td>
</tr>
</tbody>
</table>

You may only write to a user now and you should close down the subsystem as soon as possible. Generally this means the operator wants to shut down the Operating System.

9 Something inconsistent or incorrect has occurred. You should AMEND with a dump so that it is possible to ascertain the cause of failure.

10 MILTEN is not up.

11 You have made a bad call (example: MILTIE (100))

12 MILTIE has gotten invalid data from MILTEN.

13 An Up-leveled command was not a legal MILTEN command.

14 unused

15 Idle attention hit during processing -- i.e., at a time when no input or output was in progress, a user hit ATTN. This will cause failure for the next call to MILTIE; however, the operation requested will first be performed. If that operation would cause failure, the idle attention will be lost, and the error-code will pertain to the latter operation failure.
If the previous call was successful, then use of MILTIE (35) is inappropriate, and will signal failure (error 11). Otherwise it will always be successful.

Getting partial results after ATTN

If the user terminates prompt-text or read-text by hitting ATTN, the subsystem may learn at what point ATTN was hit.

(SNOBOL4) partial-text = MILTIE (37)

(PL/I) CALL MILTIE (37, dummy, dummy, partial-text, error-code);

The text returned will be the truncated prompt-text if ATTN was hit during prompt. If ATTN is hit during read, partial-text is the prompt-text followed by the read-text which was entered.

If MILTIE (37) is used and ATTN was not hit on the preceding Converse, failure will be signalled. A MILTIE (35) will indicate error 11 (bad call to MILTIE) to indicate this error.

Incorporating MILTIE into SNOBOL4

The statement:

LOAD('MILTIE(INTEGER, STRING, STRING)STRING')

must be executed to link MILTIE into a SNOBOL4 program. A JCL card of the form:

//SNOLIB DD ...load-mods...

must be supplied. Load-mods is a partitioned data set containing a member named MILTIE which is a load module (not an object module). You should obtain a copy of the source and assemble and link-edit it into your own library. A version of MILTIE is present in the author's library, but there is no guarantee it will always remain there.

Incorporating MILTIE into PL/I

The same comments concerning getting your own copy of source code apply to the PL/I version.

You must declare MILTIE in PL/I as follows:

DECLARE MILTIE ENTRY(FIXED BIN(15), CHAR(*) VAR, CHAR(*) VAR, CHAR(*) VAR, FIXED BIN(15));

Note: MILTIE runs with Version 5 of PL/I(P) and not under Version 4.

Appropriate JCL to include MILTIE requires something resembling the following:

//LKED.USER DD ...load-mods...
//LKED.SYSIN DD *
INCLUDE USER(MILTIE)
/*

Load-mods is a partitioned data set consisting of load-modules with MILTIE as a member load module, or a PDS (containing only object modules) with MILTIE in object module form.
Acknowledgements

I wish to thank John Steffani for his yeoman effort in doing the coding of MILPET in its SNOBOL4 version. Frank Rothacker ably met the challenge of enabling PL/I users to utilize this package. No thanks at all to MILTMN himself who forced us to all this effort!