Inter-Office Memorandum

To	Alto Users	Date	July 9, 1982
From	Martin Newell, Beau Sheil, Peter Deutsch, Dan Swinehart	Subject	IF - Conditional execution of commands
Filed on	[Maxc/Indigo]<AltoDocs>If.press		

XEROX

INTRODUCTION

The program *If* implements conditional execution of groups of commands to the Alto Executive. The conditions involve the existence, lengths, and read/write dates of files, as well as string matching on the contents of files. The original motivation for If was its use in conjunction with the Bravo macros for compiling BCPL and MESA programs, though other uses have become apparent.

Example

The Mesa compiler creates an .errlog file only if there were errors. Thus, a command file for compiling a Mesa program, prog.mesa, then entering Bravo only if there were errors is:

```
Delete prog.errlog
Compiler -p/c prog.mesa
If prog.errlog then Bravo/m prog
```

This is a modification of the m quit macro in bravo for MESA. Also see Examples at the end of this document.

Where to find If

If is on: [Maxc]<Alto>If.run

SPECIFICATION

Syntax

```
If <condition>[ then <commands>][ else <commands>]
```

where [ ] denotes an optional clause, and `<condition>` is one of:

`filename`	(yields true if file exists)
`filename/l=N`	(yields true if file exists and length = N)
`filename/l<N`	(yields true if file exists and length < N)
`filename/l>N`	(yields true if file exists and length > N)
`filename/s<string>`	(yields true if file exists and contains <string>)
filename/r=otherfilename/r	(yields true if both files exist and have equal read dates)
{similarly for filename/r<otherfilename/r, etc.}	
{similarly with either or both /r changed to /w, referring to write date}	
{similarly with either or both /r changed to /c, referring to creation date}	

N is a decimal constant, and <string> is a string delimited by whatever character immediately follows the switch /s, e.g. `filename/s\some text\`.

`<commands>` is any list of Alto executive commands separated by semicolons.

Note that it is necessary to escape semicolons past the Executive command line reader.

It is possible to nest Ifs by enclosing <commands> in braces ('{' and '}'): the braces will be removed before the <commands> are passed to the Executive, but occurrences of " then " or " else " within the braces will not terminate the <commands> with respect to the outer If.

Switches, "then", "else", and string matching all treat upper and lower case letters as equivalent.

Semantics

In the case that <condition> yields true the <commands> in the then clause (if present) are passed to the Executive (via rem.cm) for execution, otherwise the <commands> in the else clause (if present) are passed.

In the case of the /s switch each occurence of the character * in the <string> indicates a wildcard to be matched by zero or more characters in the file. This also has to be escaped past the Executive command line reader.

EXAMPLES

1. A quit macro in user.cm for achieving the example given in the Introduction is:

```
M.QUIT:"*q
Delete @4aᶜaᶜaᶜaᶜerrlog
Compiler -p/c @4 If.run/r @4aᶜaᶜaᶜaᶜerrlog then Bravo/m @4aᶜaᶜaᶜaᶜaᶜ
"
```

   This uses the MESA run switch /r.

2. The equivalent of the example given in the Introduction for the BCPL world is:

```
bcpl/f prog.bcpl
If prog.bt/s:ERROR: then Bravo/b prog
```

   String matching is needed since BCPL creates a .bt file whether or not there are errors.

   A quit macro in user.cm for achieving the above is:

```
B.QUIT:"*q
bcpl/f @4
If @4aᶜaᶜaᶜaᶜbt/s:ERROR: then Bravo/b @4aᶜaᶜaᶜaᶜaᶜ
"
```

3. An extension of the above example to both load and run a successfully compiled program is:

```
bcpl/f prog.bcpl
If prog.bt/s:ERROR: then bravo/b prog ^
                  else bldr prog'; prog
```

4. A further extension, which compiles the program only if it has been edited since last being compiled, is:

```
If prog.bcpl/c>prog.br/c then bcpl/f prog.bcpl
If prog.bt/s:ERROR: then bravo/b prog ^
                  else bldr prog'; prog
```

5. A command file to edit file.bravo which may be held either locally or remotely:

```
If file.bravo else Ftp Ivy retrieve/c file.bravo
Bravo/n file.bravo
```

6. Of course the above example is good only for the specific file file.bravo. It makes better sense in conjunction with the system Do.run (q.v.):

```
If #1 else Ftp Ivy retrieve/c #1
Bravo/n #1
```

   which can be used with a command of the form:

```
Do edit file.bravo
```