Inter-Office Memorandum

To	Alto Users	Date	December 30, 1980
From	Martin Newell, Lyle Ramshaw	Subject	Command files with parameters
Filed on	[Maxc]<AltoDocs>Do.press		

XEROX

INTRODUCTION

The program *Do* acts on a .do file as @ acts on a .cm file, except the file may contain parameters for substitution.

Example

To edit a file on a remote file server, let edit.do contain:

```
//Edit file #1 on file server #2
Ftp #2 retrieve/u #1
Bravo/n #1
Ftp #2 store/c #1
```

Then the command:

```
Do edit spec.bravo maxc
```

will generate:

```
//Edit file spec.bravo on file server maxc
Ftp maxc retrieve/u spec.bravo
Bravo/n spec.bravo
Ftp maxc store/c spec.bravo
```

Where to find Do

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

SPECIFICATION

Command line syntax is:

```
Do [<file> [param1 [param2 [... [paramN]...]]]]
```

Missing items will be requested interactively (see below). Do first substitutes the given parameters into the text of file, and then presents that text to the Exec as a command string. Multiple spaces are equivalent to a single space as a separator of items on the command line.

Parameter Specification

The *n*th parameter is substituted for each occurence of #*n* or #*n*# in <file>, the latter form being required when the parameter is to be followed by one of: {digit, #, ?, =, *}. The latter form may always be used if in doubt.

The string ## should be used to indicate a literal # whenever it is to be followed by one of: {digit, #, ?, =, *}.

In specifying the parameters, escape is the symmetric double-quote character ("); thus, "<ch> is equivalent to an uninterpreted <ch>. This is useful for including spaces in a parameter, e.g.

```
Do file.do a" parameter" with" five" internal" spaces
```

To include a symmetric double-quote itself, type two of them: "".

A null actual parameter may be specified by typing the single-character string ‘‘-’’ on the command line, or interactively to the request for that parameter. In those rare instances where you really want to specify ‘‘-’’ as an actual parameter, you can convince Do not to translate ‘‘-’’ into the null string by typing ‘‘"-’’ instead. Null parameters may also be specified interactively by typing <cr> to the request for that parameter.

Missing Parameters

Prompts: If no value for the *n*th parameter is given in the command line then the value of that parameter will normally be requested interactively, using a standard prompt. The prompt may be changed by including it with the first appearance of that parameter in the .do file, thus:

#*n*?*prompt*#

where *n* is the number of the parameter as before, and *prompt* is the required prompt string, conforming to the rules for typing parameters (i.e. using " as escape).

Defaults: Another possibility when no value for the *n*th parameter is given is to provide a default in the .do file, thus:

#*n*=*default*#

where *default* is the default value of parameter *n*, again conforming to the rules for typing parameters.

The trailing # can be omitted from a prompt or default if the next character is <space> or <cr>. Again, if in doubt then include the trailing #.

Automatic file name extension

On file lookup if <file> includes an extension then no other extension is attempted. Otherwise the extensions .do and .cm are attempted, followed by an attempt at the file name with no extension. If that fails, or if no file name was given, the user is prompted for the correct file name. Any typed parameters are retained and should not be retyped.

Nesting

Do does the right thing if embedded in a string of commands. Do's may be nested arbitrarily deep.

Mapping feature

It is often convenient to be able to perform some operation once for each file name in a list. If the first occurrence of the *n*th formal parameter in the .do file includes a trailing asterisk, as in #*n**, then Do will instantiate the .do file multiple times, first substituting the *n*th actual parameter for the *n*th formal parameter, and then the (*n*+1)st actual for the *n*th formal, and so on until the actual parameters are exhausted. Only the highest numbered formal parameter may be mapped in this way. Note that it is impossible to default any parameters when using this mapping feature.

A couple of hacks

The ^U command in the Executive can be used to create a special purpose .do file in Line.Cm with only a few keystrokes: for details on ^U, see the Executive documentation. Also note that, when typing a parameter interactively, it is perfectly legal to supply the string ''//''; this can cause wondrous things to happen later, since it causes the Executive to ignore the rest of the line on which it appears.

EXAMPLE 1

As another version of the example given at the beginning, but using a prompt and a default, let edit.do contain:

```
//Edit file #1?Edit" file:# on file server #2=Ivy#
Ftp #2 retrieve/u #1
Bravo/n #1
Ftp #2 store/c #1
```

[Note: neither of the trailing #'s above is necessary - they were included for clarity]

1. The command:

```
Do edit spec.bravo maxc
```

will generate (the same as in the initial example since all parameters are given):

```
//Edit file spec.bravo on file server maxc
Ftp maxc retrieve/u spec.bravo
Bravo/n spec.bravo
Ftp maxc store/c spec.bravo
```

2. The command:

```
Do
```

will provoke the questions (and possible answers):

```
Do file: edit
Edit file: spec.bravo
```

and generate:

```
//Edit file spec.bravo on file server Ivy
Ftp Ivy retrieve/u spec.bravo
Bravo/n spec.bravo
Ftp Ivy store/c spec.bravo
```

3. Usually the command will be somewhere between these two extremes, making use of the defaulting, e.g.

```
Do edit spec.bravo
```

which would generate the same command file as in the previous example, but without any prompting.

EXAMPLE 2

Here is an example that uses the mapping feature. Suppose that your Bravo is configured so that, when invoked in the manner

```
Bravo/H spec.bravo 3
```

it will print three copies of spec.bravo and then quit. You can use Do to help you print many files by putting the following text into the file hardcopy.do:

```
//Hardcopy #1 copies of #2* first
Bravo/H #2 #1
//then go on and do the rest
```

Then, giving the Executive the command:

```
Do Hardcopy 5 *.bravo
```

will print five copies of all of your .bravo files.