CHAIN Statement

CHAIN *strex*
CHAIN *strex*, **RETURN**
CHAIN *strex arglist$*
CHAIN *strex arglist$,* **RETURN**

The CHAIN statement stops the current program and starts up the program in the file named in *strex*. If the target program is not accessible, an exception does NOT occur. Instead a black DOS type SYSTEM COMMAND window opens with a white default OUTPUT window and True BASIC crashes. To avoid this happening you are advised to first check that *strex* exists BEFORE attempting to CHAIN to it.

IF *arglist$* is present then there MUST be a matching PROGRAM statement in the target program with the same number of variables as *arglist$.* IF the PROGRAM statement is missing or the number of variables does not match *arglist$* then an exception does NOT occur. Instead the *arglist$* is ignored if the PROGRAM statement is missing, or if the number of variables is incorrect then only those that are correct will be filled. For example if there are 3 parameters in *arglist$* and only 2 in the PROGRAM parameters then the first two from *arglist$* will be transferred to the PROGRAM parameters.

The parameters in *arglist$* consist of variables separated by spaces, or one string variable that includes spaces in the text, e.g.

```
LET arglist$ = “Mary had a little lamb” (5 parameters)
LET arglist$ = arg1$ & “ “ & arg2$ & “ “ & arg3$ (3 parameters)
```

If the individual parameter is a filename then it is likely the filename may contain spaces. CHAIN considers such spaces as extra parameters, e.g.

```
LET arglist$=”C:\Program files\myfolder\myfile”
```

This will count as 2 parameters “C:\Program” and “files\myfolder\myfile”
To avoid such problems you may use the following two sub-routines.
The parameter passing mechanism in CHAIN is by value, the same as defined functions.

This routine checks to see if the target program exists, and cleans up the filename *argfile$* that may be present in *arglist$.*

```
SUB check_CHAIN(targetfile$,argfile$)
    WHEN EXCEPTION IN
       OPEN #99:NAME targetfile$,create old,access outin,org byte
       ASK #99: FILESIZE bytes
       LET error$=””
       CLOSE #99
    USE
       CLOSE #99
       CAUSE EXCEPTION 10005
```

```
        EXIT SUB
    END WHEN

    LET length=len(argfile$)
    FOR n=1 to length
        IF argfile$[n:n]=chr$(32) then
            LET argfile$[n:n]=chr$(31)
        END IF
    NEXT n
END SUB
```

This routine should be used in the target file to restore the original filename.

```
SUB restore_name(argfile$)
    LET length=len(argfile$)
    FOR n=1 to length
        IF argfile$[n:n]=chr$(31) then
            LET argfile$[n:n]=chr$(32)
        END IF
    NEXT n
END SUB
```

If the RETURN clause is missing, all memory storage associated with the parent program is released, allowing the target program to occupy and use more memory.
If the RETURN clause is absent, any runtime error that occurs in the target program but is not handled within the target program by WHEN EXCETION IN will be handled and reported by the system.

If the RETURN clause is present, the parent program is retained in memory, and when the target program finishes, control is returned to the parent program at the first statement following the CHAIN statement.
If the RETURN clause is present, any runtime error that occurs in the target program but is not handled within the target program by WHEN EXCEPTION IN will be sent back to the parent program. If the target program is in source form and contains syntax errors, the exception number will be 10005, but the message will describe the actual error.

If the string-expression after the word CHAIN begins with a “!”, the rest of it will be taken as a command to the operating system. If the string-expression begins with a “!&”, and a RETURN clause is present, the parent program will continue immediately; and the command will be executed in background.

NOTE: On some Windows systems, "!" and "!&" behave effectively the same. There will still be a small difference, as the latter will return immediately, while the former will wait until the CHAINed application is launched before returning.

When the target program starts, all modules associated with it are initialised, even if the target program has already been chained to previously. However, loaded modules are not re-initialized.

On personal computer systems, chaining from source or compiled programs is permitted to either source or compiled target programs. Chaining to and from executable (bound) programs is also permitted except there are restrictions with UNIX based operating systems.

A target program can itself chain to another program. This process can continue indefinitely and is limited only the available memory.