Program Cache

This paper discusses Program Cache, which was introduced in version 3.1 of dL4 for Unix.

A single memory image of a program or library can be shared between different processes and users by using a shared program cache. This feature, available only on Unix systems, can greatly reduce the amount of memory needed to support multiple users accessing large dL4 programs. Using the cache is largely transparent to both users and applications. Cached program files are accessed using normal program paths and obey the normal rules for lookup and access permission. Programs can be modified and re-SAVEd while the cache is active without disrupting other users. Any users executing the older version of the program from the cache will continue to execute that older version while new users will invoke the most current version.

Using the program cache does not require any programming changes in applications. Programs saved under pre-3.1 versions of dL4 must, however, be reSAVEd (or LOADSAVEd) to use the new program file format. Programs in the pre-3.1 program file format can be executed, but they will not be placed in the program cache or shared between users.

The program cache is enabled and configured using the new environment variable DL4CACHE. The value of DL4CACHE is a file specification of the form:

"<access-permissions> [size] name"

where:

"<access-permissions>" is a standard dL4 file access option such as "<644>" or "<W>". If ?[size]? is specified, then the permissions will be treated as BUILD permissions, otherwise they will be used as OPEN permissions. "<access-permissions>" is an optional value.

"[size]" specifies the size of the program cache as a number of records and a record length in bytes similar to that used for contiguous files. For example, "[256:1024]" specifies a 256 kb cache. Note that a large cache will only consume virtual memory and does not reserve physical memory. If a "[size]" value is specified, the cache will be created if it does not exist. If "[size]" is not specified and the cache does not exist, then no cache will be used.

"name" is the name of the Unix semaphore and shared memory resources used by the program cache. Any decimal ("nnn"), octal ("0nnn"), or hexadecimal ("0xnnnn") format name will directly converted to a Unix resource id value (as displayed by the Unix "ipcs" utility). Any other name will be used as a seed to generate a pseudo-

random resource name. The standard cache name of "0xdddc0500" should be used unless this value conflicts with other applications or it is desired to maintain several different caches for different groups of users. "name" is a required value.

Example:

```
DL4CACHE="<666> [2048:4096] 0xdddc0500" export DL4CACHE
```

If the value of DL4CACHE is illegal or if the cache cannot be accessed, caching will be disabled. The status of the cache can be determined by using mode 0 of the ProgramCache() intrinsic as shown in the "List entries in cache" example shown in the ProgramCache() description later in this paper.

In order to use a shared program cache, the operating system must be configured to support both shared memory and semaphores. On many Unix systems, the default maximum size for shared memory will need to be increased. Please see your operating system documentation for instructions on how to configure shared memory and semaphores.

A program cache is created by the first user that enters dL4 with a DL4CACHE value that specifies a cache size and specifies a cache name that does not exist. Once created, a program cache persists until deleted by the ProgramCache() intrinsic (described later in this paper) or the operating system is reloaded. The program cache can also be deleted manually by using the Unix "ipcrm" utility to remove the shared memory and semaphore ids used by the cache.

Each user accesses the program cache in either the dynamic or the static mode. The cache mode is dynamic if a user has write access to the program cache and static if the user has read-only access to the program cache. Read and write access is controlled by the access permissions specified in the DL4CACHE environment variable (see above). Note that the mode is specific to the user and different users can be setup to use the cache in different modes.

If the user's cache mode is dynamic, all programs and libraries are entered automatically into the cache when they are used. If a new program or library is invoked and the cache is full, programs and libraries that have no current users will be deleted from the cache until sufficient space is available. If sufficient space cannot be freed, the new program or library will be loaded into the user's private memory. A typical DL4CACHE value for dynamic mode use is "<666> [16384:1024] 0xdddc0500". This provides a 16 megabyte cache with write access permitted to everyone. A larger or smaller cache can be used depending on the number and size of frequently used programs.

The cache is used in a static mode whenever a user lacks write access to the cache. In static mode, the user never enters programs or libraries into the cache. Programs and libraries are loaded into the user's private memory unless a copy of the

program file has been loaded permanently into the program cache by a user in dynamic mode via the ProgramCache() intrinsic (described in a later section of this paper). Static mode has two very important advantages: it avoids thrashing and offers higher security. A cache used in static mode cannot be corrupted either accidentally or deliberately by a user.

Using a cache in static mode is more secure, but it is also more complex. The cache must be created and initialized in dynamic mode before the static mode users enter dL4. For example, suppose a system has two megabytes of frequently used dL4 libraries. At system startup time, a dL4 process would be run with a DL4CACHE value of "<644> [2500:1024] 0xdddc0500". The process would use the new ProgramCache() intrinsic (see below) to add each of the frequently used dL4 library programs to the cache. Other users would then be started in dL4 with a DL4CACHE value of "<W> 0xdddc0500" which provides read-only (static) access.

A new standard intrinsic CALL, ProgramCache(), has been added to dL4. The intrinsic procedure ProgramCache() is used to read the current shared program cache status and to manipulate the cache. An error will be generated if improper arguments or argument values are passed to ProgramCache(). Any error that occurs while processing the operation will be reported by setting the error code argument to a non-zero dL4 error code.

BASIC syntax:

```
Call ProgramCache(0, errorcode, position, filename, usagecount)

Call ProgramCache(1, errorcode, filename)

Call ProgramCache(2, errorcode)
```

Mode 0 - Read next entry in cache.

Mode 1 - Load program into cache as a permanent entry.

Mode 2 - Delete cache when the current process exits.

Where:

errorcode -

a numeric variable that will be set to 0 if the operation is successful or to a standard dL4 error code if not. For example, if the cache is not available, the statement

```
Call ProgramCache(0,e,p,f$,c)
```

will set the variable "e" to 42 (file not found).

position - a numeric variable that determines which cache entry is read. "position" should be set to zero to read the first entry. Each mode 0 call will update the value of "position" so that the next call will read the next cache entry. The precision of "position" must be such that it can contain any value between 0 and 2^32-1 without any loss of precision (a 3% variable is adequate). The caller should only pass "position" values of zero or those returned by the previous mode 0 call to ProgramCache().

filename - a string variable or expression that will receive a program file path (mode 0) or supply a program file path (mode 1).

usagecount - a numeric variable set to the number of users of the program. A usage count of -1 indicates that the program has been added to the cache as a permanent entry.

Example: Adding a program to the cache as a permanent entry

```
Declare Intrinsic Sub ProgramCache
Dim 1%, ErrorCode
Call ProgramCache(1, ErrorCode, "MenuLibrary.lib")
```

Users in static cache mode can only use cached programs and libraries that have been added as permanent entries. These permanent entries must be created by a user in dynamic cache mode using mode 1 of ProgramCache(). Once made, permanent entries cannot be individually deleted because there is no way to determine whether or not a static mode user is currently executing the program or library. See the program cache description above for more information on dynamic and static cache modes.

Example: List entries in cache

```
Declare Intrinsic Sub ProgramCache
Dim 1%, ErrorCode, 3%, CachePos, File$[200], Usage
CachePos = 0
Do

   Call ProgramCache(0, ErrorCode, CachePos, File$, Usage)

   If ErrorCode Exit Do

   If Usage < 0
```

```
      Print "Permanent ";

    Else


      Print Using "######### ";Usage;

    End If

    Print File$
  Loop
  If ErrorCode = 73 Print "The program cache is not enabled"
```

Note: If a cached program is edited and resaved, then the same program name is listed multiple times.

Example: Deleting the program cache

```
  Declare Intrinsic Sub ProgramCache
  Dim 1%, ErrorCode
  Call ProgramCache(2, ErrorCode)
```

This example will delete the program cache when the current user exits dL4. The program cache should be deleted if it is desired to increase the size of the cache or if the cache has become corrupted. The cache can be deleted only by the owner of the cache or by the root user. Since the cache cannot be deleted until the user exits, no error is returned if the caller lacks delete permission. All other users should exit dL4 before the cache is deleted.