Published: 03/28/69

Identification

the epl_daemon
V. L. Voydock

Purpose

The epl daemon is a daemon process which will perform all epl compilations and eplbsa assemblies on Multics. This is done to prevent the overloading of the system which would occur if a number of processes tried to perform compilations simultaneously, and to preserve drum space (since both epl and eplbsa are impure).

Usage

The user types the command "epl alpha [options]", ("eplbsa alpha [options]") just as if he were invoking the epl (eplbsa) command. (See MSPM sections BX.7.03-BX.7.04 for a description of these commands). The only restriction is that alpha must be an entry name in the user's working directory. When control returns to the user he may do anything he wishes (issue other commands, logout, etc.). At some future time when the daemon finishes processing the user's request the output segments produced by the epl (eplbsa) commands will be placed in the directory which was the user's working directory at the time the command "epl alpha" was typed. (Here after we will call this directory "the user's working directory"). Messages from the daemon and the compiler (assembler) will be appended to the file "epl_daemon.error" in the user's working directory (if this file does not exist it will be created).

Implementation

Compilations are signalled in the following manner. When the user types "epl alpha" ("eplbsa alpha") the procedure epl creates a unique bit string "x" and from x a unique character string "uchar". It creates a link with entry name "uchar" in the daemon's directory whose associated path name is of the form "wdir>alpha.i" where wdir is the pathname of the user's current working directory, and i=1 if "epl alpha" was typed and i=2 if "eplbsa alpha" was typed. It then picks up the daemon's process identifier and the name of the event channel along which the daemon

is to be signalled from the segment "daemon_info". [This information was placed in daemon_info by the daemon initializer "epl_daemon_init". (see below)]. Using these it signals the daemon (via IPC) sending "x || opt" as the event message, where opt = "00"b if no listing is desired and opt = "01"b otherwise.

The daemon upon awakening behaves as follows:

1. picks up event message, masks last two bits and forms entry name of the link to the file to be compiled using unique_chars.

2. gets path name from entry name using ufo$chase (see BY.2.01).

3. picks working directory of caller off of pathname. (This is the reason for the restriction of arguments to links or branches in the caller's working directory).

4. switches to this working directory.

5. Changes the output stream "user_output" to write into epl_daemon.error (in user's working directory).

6. finds out (from last character of the pathname obtained from ufo$chase) whether epl or eplbsa should be called and makes corresponding call with the options specified.

7. prints timing information for compilation into epl_daemon.error.

8. attaches the stream "user_output" to user_i/o.

9. removes link to alpha and changes back to its own working directory.

10. returns to wait coordinator.

During system initialization a program "epl_daemon_init" is called to (oddly enough) initialize the epl daemon. It creates a directory "epl_daemon_dir" (if this directory doesn't already exist) and creates an event-call channel over which messages will be sent to the daemon. It places the name of this channel together with the process identifier of the daemon into the segment "daemon_info", a data base containing information about all daemon processes. Finally it signals the daemon (via IPC) once for every link presently

in the daemon's directory. [This causes any compilation (assembly) requests left over from the last system shutdown to be processed since links aren't removed until compilation (assembly) of a file has been attempted.]

The use of daemon processes in Multics brings up a number of access problems. These problems and possible solutions for them will be the topic of another document.