To: MTB Distribution

From: David Spector

Date: 14 May 1979

Subject: New Program Coordination Facility

Motivation

Most Multics system programmers have wished at times for a mechanism to coordinate the changes they make to the system, in order to avoid discovering too late that someone else has been working on the same programs (or include files, bind files, etc.) at the same time. It is appropriate to solve the problem using an online procedure since Multics excels at other such interactive applications (memos, mail, lister files, etc.).

Program Opening

The most natural mechanism to provide the desired interprogrammer coordination is the concept of opening and closing programs. A program is said to be open to (or by) a programmer when the programmer has reserved it with the intention of modifying a local copy of the program and later installing the program as a standard part of Multics. An open program is closed when the programmer frees it so that other programmers can open it, usually because it has been submitted for installation.

Administrative Control

A facility for supporting opening and closing of programs can implement various degrees of administrative control. At one extreme is a facility that disallows access to any system program source segment except when authorized by its policies for contention resolution. At the other extreme is a facility that serves an advisory function only, leaving enforcement completely up to the individual programmers. Without entering into a lengthy discussion of the relative merits of these two extremes, it is clear that the second, advisory, facility is by far easier and quicker to implement; it is this type of facility that is the subject of this MTB.

Requirements

A good program coordination facility should support a variety of features. It should allow opening and closing of programs as described above. It should allow the association of programmers as the maintainers (as distinct from changers) of programs, independently of whether or not they have the programs open for change. More than one programmer should be permitted to have a given program open at the same time, providing that each is warned of the fact that other programmers have the program open at the time they attempt to open it, and providing the other programmers are informed (via interactive messages) of the new opening. Opening a program as maintainer, on the other hand, should be allowed only to one person at a time; this restriction simplifies the notion of who is responsible for a program.

Opening, closing, and maintaining programs should create database entries recording the date and time of the change, the group_id of the process making the change, and an optional comment field provided by the programmer to describe the changes being made. This comment field can include the pathname of the directory in which a local copy being changed resides, and can be used to fulfil other site or departmental conventions and standards. A command should be available to examine the status of a program (or of all programs, alphabetically sorted), without causing the program to be opened or closed. Administrative commands should be available to correct erroneous entries and to control access to the database.

Universality

By designing a generalized facility, we hopefully have produced a product useable in environments considerably different from those found at the Phoenix and Cambridge software development sites. Indeed, there is no requirement that the environment being coordinated even exist on the computer system that implements the facility. Thus a site can coordinate changes being made to programs that are stored, edited, and compiled on systems other than their Multics system. Multiple databases can be created, if desired, to separate the coordination of programs in multiple environments.

In-House Use

The facility described above has been implemented using MRDS at both MIT and Phoenix (System M). The commands and help files (info segs) are located in the directory >udd>m>ds>l (l for 'library'). They are described below.

04/25/79 open_program

Syntax: open_program program_name {-control_args}

Function: The open_program command reserves a named program to the user in order to coordinate software development activities.

Arguments:
program_name
Entryname of program to be opened. The language suffix must be included.

Control arguments:
-maintainer, -mn
User is to be maintainer of the program. This is separate from opening the program in order to change it.
-comment STR, -cm STR
Associates the string STR with this opening as a comment.
-long, -lg
Causes more info to be displayed when other users have already opened the specified program.

Notes: To close programs, use the close_program command. To display the status of a program, use the status_program command.

The program_name should be the primary source name of the program or component (if a bound segment). No checking is done by open_program.

Several people can open a program at the same time but only one person can be its maintainer.

Opening a program already opened or maintained by others causes them to be notified via an interactive message.

Comment strings are 128 chars max. If they contain embedded blanks, they must be quoted.

The database used is open.db in the same directory as the command itself. The database is a directory (MRDS database).

Examples:
open_program list.pl1 -cm "Adding -special control arg."
This causes list.pl1 to be opened for changing.
close_program list.pl1 -cm "Source is in >udd>m>ds>list."
This causes list.pl1 to be closed for changing.
status_program list.pl1
This displays status of list.pl1.

04/25/79 close_program

Syntax: close_program program_name {-control_args}

Function: The close_program command releases a named program that was reserved by using the open_program command. It is used to coordinate software development activities.

Arguments:
program_name
Entryname of program to be closed. The language suffix must be included.

Control arguments:
-maintainer, -mn
User is currently maintainer of the program. This is separate from closing a program that was being changed.
-comment STR, -cm STR
Associates the string STR with this closing as a comment. This overwrites the comment, if any, associated with the opening.
-long, -lg
Causes more info to be displayed when other users have the program open.
-delete, -dl
This control arg allows correction of mistakes by querying and allowing deletion of any or all database entries for the named program (-mn can be specified).

Notes: To open programs, use the open_program command. To display the status of a program, use the status_program command.

The program_name should be the primary source name of the program or component (if a bound segment). No checking is done by close_program.

Closing a program that is opened or maintained by others causes them to be notified via an interactive message.

Comment strings are 128 chars max. If they contain embedded blanks they must be quoted.

04/25/79 status_program

Syntax: status_program {program_name} {-control_args}

Function: The status_program command displays the open/closed/maintained status of one or more programs. It is used to help coordinate software development activities.

Arguments:
program_name
Entryname of program to be opened. The language suffix must be included. A program name is not given if the -my or -all control arguments are given.

Control arguments:
-long, -lg
Causes more info to be displayed, including entire open/close/maintain history of the program(s).
-my
Displays status of all programs opened or maintained by the user.
-all, -a
Displays status of all known programs.

Notes: To open programs, use the open_program command. To close programs, use the close_program command.

The program_name should be the primary source name of the program or component (if a bound segment). No checking is done by status_program.