TO: Distribution

FROM: Gary C. Dixon

DATE: January 23, 1972

SUBJECT: Plan for Multics System Library conversion and for shifting Library maintenance to the 6180.

During the latter part of February and the month of March, we plan to change the organization of the Multics Online Libraries to the structure documented in Appendix 1 of this memo. This structure will simplify Library maintenance procedures, speed up Library searches and Library installations, and increase the reliability of the Online Libraries.

Also during this period, we plan to shift maintenance activities for the Multics System Libraries (including the Hardcore Library) to the 6180, and to move the PDO Multics Support Group headquarters from Bldg NE43 (545 Tech Square) to Bldg 39. After the shift is complete, only EMERGENCY BUG FIXES will be installed into the Multics System Libraries on the 645.

The time table for the conversion and shift operations is shown below. Please note that these dates are susceptible to postponement because they depend upon a number of uncontrollable factors.

Phase	Date	Description
1	02/19/73	- begin moratorium on Online Library installations on the 645. - begin conversion of Multics Online Libraries.
2	03/05/73	- finish conversion of Multics Online Libraries. - end installation moratorium; start Online Library installations on 645.
3	03/09/73	- stop installations on 645, except for EMERGENCY BUG FIXES. - move Multics System Libraries to 6180. - take over Multics System Library maintenance functions on the 6180.

Phase	Date	Description
4	03/12/73	- begin performing Multics System Library installations on the 6180. - move Online Library installation headquarters from NE43-504 to 39-411. - move Hardcore Library installation headquarters from NE43-505 to Bldg 39. - move Multics System Library listings from NE43-504A to 39-413. - install Multics System Library microfiche in NE43-504A.

During the Library conversion phase, there may be short periods when a particular Online Library is not available to system programmers. To reduce these periods of unavailability, we plan to convert one Online Library at a time to the new organization, rather than converting all at once.

System programmers will still be able to extract segments from each Online Library after it has been converted. get_library_source is already capable of extracting segments from both the old and the new Library organizations. Therefore, the Library reorganization will be transparent to gls users.

The msl_global_format, msl_short_format, and msl_info programs, which provide information summaries about the contents of the Libraries, will be replaced in the near future by the library_list and library_info commands. The proposed calling sequences for these two commands will be available in an info segment when the commands are installed.

Appendix 1: The New Library Organization

This appendix presents a preliminary draft of two sections of a new manual, the MPM Library Maintenance Supplement. These sections describe the new Library organization which is planned for the Online Libraries in the near future. They reflect the eventual Library configuration, as it will appear after the Version 1 PL/I compiler has been phased out. Until this phase out occurs, however, we plan to retain the Version 1 PL/I Library, >library_dir_dir>pl1, and to convert it to the new Library organization if such a conversion is feasible.

INTRODUCTION TO THE MULTICS SYSTEM LIBRARIES

At the heart of the Multics System are the Multics System Libraries. The procedure and data segments in these Libraries prepare the system for operation at bootload (system startup) time, control and monitor system operation, and maintain the Multics Storage System hierarchy.

Each of the Multics System Libraries may be classified as either an Online Library or an Offline Library, according to the medium used to store the segments of that Library. The segments of Online Libraries are stored in the Multics Storage System. The procedure and data segments of Offline Libraries are stored on tape or cards which can be bootloaded. Offline Library segments are stored outside of the Multics Storage System so that they can be used when Multics is not in operation and the Multics Storage System cannot be accessed.

The next two sections describe the contents and organization of these two groups of Libraries.

c Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc. (END)

THE ONLINE LIBRARIES

Contents

We can itemize the contents of the Multics Online Libraries formally by listing the Multics Storage System pathnames of the major Library directory nodes.

> The Multics Online Libraries consist of all segments in the Multics Storage System hierarchy which branch off of the directory nodes whose pathnames are listed below.
>
> ```
> >system_library_standard
> >system_library_languages
> >system_library_tools
> >system_library_auth_maint
> >library_dir_dir>standard
> >library_dir_dir>languages
> >library_dir_dir>tools
> >library_dir_dir>auth_maint
> >library_dir_dir>include
> >documentation>info_files
> >documentation>pt_files
> ```

Organization

As the name suggests, the organizational unit of the Online Libraries is a library. The directories in the list above may be divided into four major libraries and two minor libraries.

c Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc.

The major libraries and their component directories are:

Standard Library(1)
>system_library_standard
>library_dir_dir>standard

Languages Library(2)
>system_library_languages
>library_dir_dir>languages

Tools Library
>system_library_tools
>library_dir_dir>tools

Author-Maintained Library
>system_library_auth_maint
>library_dir_dir>auth_maint

The minor libraries and their component directories are:

Include Segment Library
>library_dir_dir>include

System Documentation Library
>documentation>info_files
>documentation>pt_files

The major libraries each consist of four directories and the segments they contain. The execution directory, whose pathname for library "X" is >system_library_X, contains: 1) bound object segments; and 2) unbound object segments. These segments may be executed as procedure segments, or they may be referenced as data segments. Segments in the execution directory are referred to collectively as executable segments.

---

(1) The Standard Library was formerly divided into two libraries, the Standard Service Library, and the Development Library.

(2) The Languages Library was formerly called the PL1 Library.

c Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc.

The object directory, whose pathname for library X is >library_dir_dir>X>object, contains: 1) object archives which may be bound to produce the bound object segments in the execution directory; and 2) non-archived object segments which are backup copies of the unbound object segments in the execution directory. The segments in the object directory are referred to collectively as object segments.

The source directory, whose pathname for library X is >library_dir_dir>X>source, contains: 1) source archives whose components may be compiled to produce the components of the corresponding object archives in the object directory; and 2) non-archived source segments which may be compiled to produce corresponding non-archived object segments in the object directory. There is a one-to-one correspondence between each object archive and a source archive, between each object component which has been compiled and a source component, and between each non-archived object segment which has been compiled and a non-archived source segment. The segments in the source directory are referred to collectively as source segments.

The lists directory, whose pathname for library X is >library_dir_dir>X>lists, contains the bind listings produced when binding the object directory's object archives to produce the execution directory's bound object segments. The listings are stored as segments, one listing segment per bound object segment.

The minor libraries are logical extensions of the major libraries. Like the major libraries, the minor libraries consist of one or more directories, each of which contains segments which serve a particular function. Unlike the major libraries, however, all of the segments which serve a particular function are grouped together in a single directory, regardless of their affiliation with major libraries. It is this grouping of segments across logical library boundaries which distinguishes minor libraries from major libraries. The grouping serves two purposes: it facilitates searches for these special-purpose segments across library boundaries; and it eliminates the need to duplicate segments which may be related to more than one of the major libraries. The Include Segment Library is a minor library which contains include segments. These are pieces of source code common to several source programs in the major

c Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc.

libraries. The compilers logically include these segments in the source programs when they are compiled.

The System Documentation Library contains info segments and pt segments. These segments contain general information about the Multics system, and specific information about Multics commands and subroutines. The information can be printed on the user's terminal using the help and peruse_text commands.

Naming Conventions

The following conventions are used for naming the segments in the Online Libraries. These conventions have been adopted over a period of years to facilitate Online Library searching, and to prevent name duplications, both among Library segments, and between Library segments and user-defined segments.

In the execution directory of major libraries, each executable procedure segment, whether it is a bound object segment or an unbound object segment, has one or more single-component names. Each name which references a subroutine entry point in the object segment ends with an underscore ("_"), to prevent conflicts with user-defined subroutine names. Each name which references a command entry point in the object segment does not end in underscore, and generally references an identically-named entry point, in order to minimize the amount of typing required to invoke a command. In addition, the first name on each bound object segment is a special name which identifies the general purpose of the bound segment. Called the "primary" name of the bound segment, it is of the form, "bound_xxx_", where the first word of the name is always "bound", and the name always ends in an underscore ("_").

Examples of names which reference subroutines are hcs_, cu_, ioa_, and user_info_. Some sample command names are create, cr, delete, dl, link, and lk. Note that an abbreviated name is usually provided for command names which are longer than three characters to simplify command invocation. Examples of primary names for bound object segments are bound_user_control_, bound_qedx_, and bound_fortran_.

c Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc.

Data segments in the execution directory of a major library usually have single-component names which end in an underscore ("_"), to prevent conflicts with user-defined subroutine or data base names. However, some data bases have multi-component names, the last component of which identifies the way in which the segment is intended to be referenced. Examples of data base names are error_table_, multics.ids, and translator_absin.absin.

The object directory of each major library contains object archives and non-archived object segments. One or more object archives may be bound together to produce a given bound object segment, "bound_xxx_", in the execution directory. If only one object archive is involved in the binding, then its primary name is of the form, "bound_xxx_.archive". If more than one object archive is involved in the binding, then the primary name of each archive is of the form, "bound_xxx_.n.archive", where n is an integer. In addition to a primary name, the name of each component of an object archive is placed on the archive itself, to simplify searching for object components. For example, the names on the single archive which is bound to produce bound_qedx_ might be:

```
bound_qedx_.archive
qedx
edx_util_
get_addr_
search_file_
bound_qedx_.bind
```

while the names on the two archives which are bound to produce bound_cg_ might be:

```
bound_cg_.1.archive          bound_cg_.2.archive
adjust_ref_count             e_v
arith_op                     expmac
bound_cg_.bind               fixed_to_float
```

Non-archived object segments are backup copies of the execution directory's unbound procedure and data segments. The names on the execution directory copy of these segments are also placed on the object directory copy, to facilitate searching.

c Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc.

The source directory of each major library contains source archives and non-archived source segments. Each source archive which contains the source components for object archive, "bound_xxx_.archive", has a primary name of the form, "bound_xxx_.s.archive". Similarly, each source archive which contains the source components for object archive, "bound_xxx_.n.archive", has a primary name of the form, "bound_xxx_.n.s.archive". In addition, the name of each source component is placed on the source archive itself, to simplify searching for source components. For example, the names on the source archive which corresponds to bound_qedx_.archive might be:

```
bound_qedx_.s.archive
qedx.pl1
edx_util_.pl1
get_addr_.alm
search_file_.pl1
```

Those on the source archives corresponding to bound_cg_.1.archive and bound_cg_.2.archive might be:

```
bound_cg_.1.s.archive          bound_cg_.2.s.archive
adjust_ref_count.pl1           e_v.alm
arith_op.pl1                   expmac.pl1
                               fixed_to_float.alm
```

Each name on a non-archived object segment in the object directory which has been compiled is also placed on the corresponding non-archived source segment in the source directory, after a language suffix has been added to the name to indicate the source language of the segment.

The lists directory of each major library contains bind listings. The bind listing produced when "bound_xxx_" is bound has a name of the form, "bound_xxx_.list". Examples are bound_user_control_.list, bound_qedx_.list, and bound_fortran_.list.

The Include Segment Library contains include segments, which are pieces of source programs. Each include segment has one or more three-component names of the form, "ident.incl.lang", where ident identifies the contents of the include segment, and lang is

c Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc.

a name suffix which identifies the source language of the include segment. Examples of include segment names are sst.incl.pl1, sst.incl.alm, and dir_entry.incl.pl1.

The System Documentation Library contains info segments and pt segments. Each info segment has one or more multi-component names of the form, "ident1.info", or "ident1.ident2.info", or ... where ident1 identify the contents of the info segment. Examples of info segment names are help.info, fortran.bugs.info, and motd.info.

Each pt segment has one or more multi-component names of the form, "ident1.pt", or "ident1.ident2.pt", or ... where ident1 identify the contents of the pt segment. Examples of pt segment names are pl1.pt, pt.pt, and ready_on.pt.

Access to Online Libraries

The following section describes the access to the Online Libraries which has been granted to users. In the vast majority of cases, access to Library segments is limited to "r" or "re" modes (even for Library maintainers) in order to protect the Libraries from accidental or intentional mishap.

Most segments in the execution directory of a major library have ring brackets of 1,5,5 and an access control list (ACL) which grants "rewa" access to *.SysDaemon.* in order to facilitate segment backup, and "re" access to all other users. Gate segments into ring 1 (the Administrative Ring) have ring brackets of 1,1,5, causing these segments to execute in ring 1 when they are called as a procedure. Some special segments have ACL's which do not permit the normal user to access the segment at all, or which allow specific users to write into the segment. These segments have been created to fulfill a specific need for a small group of users. For example, the Library maintainers gain access to ring 1 in order to modify the segments in the execution directory by calling a special installation_tools_ gate, to which only they have access.

c Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc.

All segments in the source, object, and lists directory of each major library have ring brackets of 4,5,5 and ACL's which allow "r" access for Multics system developers and for Library maintainers, "rwa" access for *.SysDaemon.*, and "null" access for other users.

Segments in the Include Segment Library have ring brackets of 4,5,5 and ACL's which allow "rwa" access for *.SysDaemon.*, and "r" access for all other users.

Segments in the System Documentation Library have ring brackets of 4,5,5 and ACL's which allow "rwa" access to *.SysDaemon.*, and "r" access to all other Multics users. In addition, some info segments contain status information which is updated by users other than the Library maintainers. These users have "rwa" access to the appropriate status info segments.

c Copyright 1973, Massachusetts Institute of Technology
and Honeywell Information Systems Inc. (END)