A MULTI-TASKING OPERATING SYSTEM FOR INTERACTIVE DATA REDUCTION

R.J. Allen and J.P. Terlouw

Kapteyn Astronomical Institute
University of Groningen
9700 AV Groningen, The Netherlands

Summary:

The development of facilities for user-system communication in interactive data-processing environments has not kept pace with the enormous advances which have led to the modern versatile multiuser/multitasking operating systems. We present here an approach to the problem of improving the efficiency of this communication, and describe a program which has been developed for this purpose.

This paper was presented at the "Workshop on IUE Data Reduction", held in Vienna on November 17-19, 1980.

1. Introduction:

Over the past ten or more years there have been enormous advancements made in the versatility and complexity of the operating systems offered by computer manufacturers for use in their machines. The evolution has gone from single-user/single-task systems (e.g. PDP-9 Keyboard Monitor) via foreground/background systems (e.g. RT11) to the multi-user/multi-tasking systems currently used in modern mini and midicomputers (e.g. RSX-11M, VAX/VMS), and now even in microcomputers. The problem is that the development of the interface with the user has not kept pace with the increasing sophistication in the operating system facilities which are offered to him. When working with a multi-user/multi-tasking operating system, it is not unusual to find an experienced user running several terminals at the same time. He starts up different (perhaps even related) processes from different terminals because there is no better way available to keep track of the several things he wishes to do without becoming hopelessly confused. The operating system may allow him to initiate dozens of interrelated tasks from terminals and even from within his own running programs, but the nature of the communication between him and the operating system has not improved since the days of mechanical teletypes. The situation with the large batch-processing computers is much better: Timesharing/multi-tasking process control is the province of a special resident executive program under the direction of the computer operator who sits in front of a special terminal screen on which the executive program continuously presents the current status of all machine activity in a visually meaningful way. Directives to the resident executive program which may alter the flow of the job stream are simply not made available to the average user at a standard terminal or to any application program he may write. On the other hand, the modern multiuser/multitasking minicomputer operating systems allow essentially every user to assume the role of the computer operator, but they do not provide him with the operators console visual display and interactive keyboard which are required in order to monitor the progress of the various processes which are relevant even just to his own problem.

We describe here one approach to improving the efficiency of the

communication between the user and the computer standard operating system by inserting a program interface between them. This program communicates on the one hand with the user by formatting the character terminal screen in a way which is meaningful to him, and on the other hand with the computer operating system via calls to standard executive subroutines supplied by the computer manufacturer.

2. Requirements for the User Interface:

A program of the sort outlined above is in a good position to do a number both general and application-specific jobs for the user during his session. For example if he is using a library of tasks for interactive data reduction in a small research-group environment of the type described by Allen and Ekers elsewhere in this volume, this "Master Control Task" or "User Operating System" should do at least the following:

- respond essentially immediately to every command entered by the user on the terminal keyboard with some reply which is meaningful to him;
- present the messages from running tasks and from itself in a standard and visually meaningful way to the user terminal screen;
- keep a record of transactions between the user, the computer operating system, and the applications programs which can help the user to retrace his actions if required;
- allow frequently-used tasks to be initiated with highly abbreviated commands;
- allow for the provision of default values to application tasks requiring parameters.

In addition to these user-oriented jobs, the program could also provide task execution statistics which may be pertinent to the allocation of the programming efforts of the group.

3. HERMES:

The requirements listed above have been met with a program (the "Master Task") which essentially plays the role of a messenger between the user, his applications programs ("servant tasks") and the computer operating system (RSX-11M). This messenger function has earned the

program its name, "HERMES". The particular implementation is, of course, specific to the host computer operating system, but a summary of the facilities which HERMES offers to the user may be of more general interest.

3.1 User communication with HERMES:

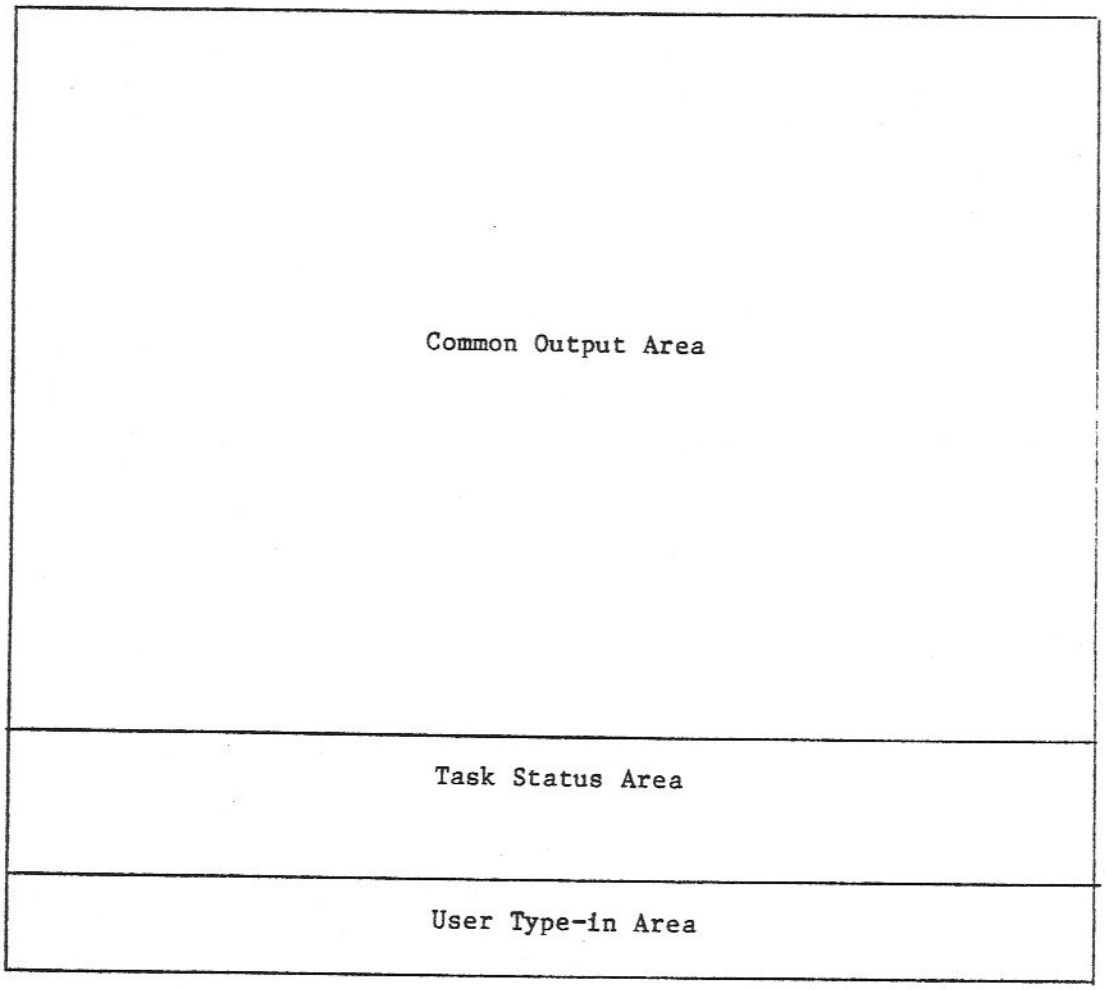
The interface to the user is via a standard cursor-addressable video display terminal. The display is an output device through which information can reach the user; the keyboard is an input device via which the user can enter information in order to "steer" his data reduction process. HERMES allows for the following kinds of user-supplied information:

- commands to run a servant task. More than one servant task can be active at one time; the maximum is determined by a parameter established when HERMES is built (currently 4);
- commands to obtain information from the system or from a servant task (e.g. ask for the current input parameters of a servant task to be displayed);
- commands that change the state of the system or the state of a servant task (e.g. abort a servant; stop the system);
- input parameters to a servant task.

Some examples of these commands will be given later. The facility also exists to permit a servant task initiated by HERMES to request ("spawn") the initiation of another servant task (also via HERMES). In this way a servant task can assume the role of the human user, passing keyword parameters to the spawned task, examining its status, etc.

3.2 Display organization:

The terminal display screen is divided into three sections, each with its own purpose, as shown in the sketch below:



1. Common Output Area (length: 18 lines)
   This section is used to display information that is sent from servant tasks and meant to reach the user via the terminal display.
2. Task Status Area (length: 4 lines)
   For every active servant task this section contains one line consisting of the taskname, and additional status information which is updated dynamically. When a servant task requires user intervention in order to continue, the status information is preceded by a blinking asterisk. In addition to this, the terminal will "beep" when a task enters such a state.
3. User Type-in Area (length: 2 lines)
   This section is used to "echo" whatever the user types.

3.2.1 The Common Output Area and the Log File:

The Common Output Area (COA) on the terminal screen is in fact one page in a "book". Pages in this book are numbered. On the right-hand edge of the Task Status Area are shown the number of the page which is being displayed on the COA, and the page number on which messages are currently being written. There are two modes of displaying information: page mode and non-page mode.
In non-page mode (the default) the current- and display-page are always the same, i.e. if a new page is initiated, the old one disappears from the screen. In page mode the information on the screen will stay there until the user instructs the system to change the page.

There are two classes of commands to manipulate the Common Output Area: control characters; and commands termined with <CR> (carriage return). The control characters always have an immediate effect, regardless of the state of any line currently being typed. These page control characters and commands do not inhibit tasks from continuing to write in the log file in the normal way.

a) Page control characters (↑ = control key)

↑P enter or leave page mode
TAB display previous page and enter page mode
LF display next page or leave page mode if highest page already displayed
↑L display highest page number ever displayed before
↑H make a hard-copy of the page being displayed plus the preceding two pages
(this command requires one free servant task entry).

b) Page commands

/HC l:m

make a hard-copy of pages l through m.

*n display page number n

*+n display page number being displayed + n

*-n display page number being displayed - n

=string or +=string

search forward for "string" and, if found, display the page containing it

-=string

search backwards for "string"

As search commands may require more time than the user is willing to wait, they are aborted when any character is typed.

3.2.2 Task Status Area:

The Task Status Area (TSA) is the middle section of the terminal screen. It consists of four lines which can contain status information of active servant tasks. To the extreme right of the TSA a time-of-day clock and the current- and display-page numbers are given. Some examples of the status information are (asterisks shown blink on the screen):

- When a task has been activated, but is not running yet;
  taskname WAITING TO BE RUN
- When a task is running;
  taskname RUNNING
  or
  taskname status message supplied by servant task
- When a task is pausing;
  taskname *PAUSING (optional message)
- When a task has itself initiated another task, and is waiting for the spawned task to finish;
  taskname WAITING FOR other taskname
- When a task requires input;
  taskname *input request message from that task
- When a task encountered an error fatal to its own execution;
  taskname errormessage -FATAL

- When a task has crashed (i.e. stopped without properly notifying HERMES);
  taskname *CRASHED
- When a task has finished processing normally;
  taskname +++FINISHED+++
- When a task has been aborted by the user;
  taskname USER ABORT

The last four messages will disappear as soon as the user initiates an other servant task or when the user types ↑R.
↑R can also be used to recover the TSA if it is accidentally destroyed.

3.2.3 User Type-in Area:

The User Type-in Area (UTA) consists of two lines located at the bottom of the display screen. 148 typing positions are available for user input. Some examples of the commands that can be typed will be described later. HERMES itself can also "type" in this area. It does so in the following cases:

1. When it is likely that a "taskname,keyword=" combination will have to be typed, HERMES types it for the user.
2. When the user types ↑C, a punched card is read and the information contained by it is typed in the UTA.
3. When the user types ↑Y or ↑T, a cursor processor is activated. When the cursor position has been read out, the coordinate values are typed in the UTA and can therefore appear as keyword parameters for input to another (active or still to be activated) task.

Text present in the UTA which is either erroneous or otherwise unwanted is simply deleted with the DELETE key (or ↑U to erase the whole line). For instance in this way a task waiting on input via a "taskname,keyword=" request in the UTA need not be satisfied before initiating some other task. ↑U erases the text, and a new command can be given by the user. If the UTA is empty, striking the space bar causes HERMES to cycle through presenting the currently unsatisfied

keywords of active tasks.
At the end of the UTA, HERMES displays error messages which are related to the line typed by the user (e.g. BAD SYNTAX, ALREADY ACTIVE, NOT PRESENT, etc.). When a line is accepted by HERMES, it will disappear from the UTA; if it is rejected (as can be seen from the error message) it stays on the display, but will disappear as soon as the user starts typing a new line. Rejected lines are not displayed in the COA and do not appear in the Log File.

4. Additional Facilities:

HERMES offers a number of other facilities which we can only summarize here:

- free format of input numbers from the user, and conversion to the particular type (integer, real, logical) required by the servant task;
- user-initiated control of task execution, such as taskname/WAIT or taskname/ABORT;
- definition of lengthy command character strings as "macros" which can be referred to later in a shorthand notation. HERMES itself makes use of this facility to store the values of keywords from the most recent execution of each task, facilitating repetitive operations;
- provision for supplying default values to tasks requesting parameter input if the user answers with carriage return instead of typing a number.

A user's manual which includes more extensive descriptions of the facilities sketched above is available from the authors.

Acknowledgements

We thank the director and staff of the Groningen University Computer Center for providing facilities and support for the work described in this paper, and our colleagues for their critical evaluation of the performance of the software.