

The Atlas scheduling system

By D. J. Howarth, P. D. Jones and M. T. Wyld

The Atlas computing system involves an operating system which is implemented by the Supervisor program held in the store of the computer. One part of the Supervisor is the scheduler program which determines the order in which available jobs are assembled in store, compiled and executed. The essential features of the operating system are outlined and the task of the scheduler program within the framework of this system is described. The advantages of a computer-organized scheduling system are pointed out, and an account is given of the salient features of the particular scheduling system which is used on Atlas.

Introduction

Atlas is the name given to a comprehensive computer system which has its origin in the Computing Machine Laboratory under Professor T. Kilburn at Manchester University. The final design is the result of close collaboration between this group and Ferranti Ltd. This paper describes the basic principles of the scheduling system adopted for Atlas. The main features discussed are the ordering of a queue of jobs awaiting execution, the allocation of main store space and tape decks, the assembly and preparation of jobs for execution, and the communication system between operator and scheduler program, which includes the allocation of external priorities to jobs. A later paper will describe the dynamic time-sharing of jobs during the course of their execution, and associated simulation studies.

Virtually no information has been published which is of practical value in formulating the Atlas scheduling system, and hence the system is designed to facilitate alterations in the light of operating experience. The basis of the system may be expected to require little changing, but certain decisions, which depend upon a balance of various factors, may require modification when the relative importance of these factors is known in more detail.

Summary of the Atlas Operating System

The principles of the Atlas Operating System, all the activities of which are controlled by a program called the *supervisor*, have been described in previous papers (1, 2 and 3). In order to appreciate the action of the scheduler, it is advantageous to summarize first the salient features of the system. It is designed to overlap the operation of those parts of the computing system which can function concurrently, namely, input and output peripherals, magnetic tapes, the central computer, and the human operators. In order to achieve overlap of input and output operations with computing, regions of the core and drum store are used as "wells," which can be filled and emptied at peripheral speeds by the peripheral equipment, and at computer speeds, for short bursts, by the central computer. These wells are supplemented by system magnetic tapes (a system magnetic tape is one which is controlled by the supervisor, in contrast to a private magnetic tape which is under direct

control of an ordinary program), which are not used to supply the central computer directly, but which are used to fill and empty the wells in core and drum store. Jobs whose peripheral input is complete are assembled in main store, prior to execution, from the system input tape or from other system tapes on which they have been loaded previously. Program results occupy the output well in main store before being recorded on the system output tape.

Complete jobs may consist of several separate input documents (a document is a self-contained section of input information presented to the computer consecutively through one input channel), one of which must contain a job description. This lists all other input documents required, the titles of any magnetic tapes required, the output streams and the type of peripheral required for each, and also supplies approximate estimates of storage space (in combined drum and core store) required for execution, computing time, and quantity of output for each output stream. These estimates are used by the supervisor program as a guard against serious program error and are also available for use by the scheduler. Decisions called for by the operator are thereby reduced to a minimum, sufficient information being supplied to the computer by the user to enable a job to be run. Allowance is made for operator intervention in exceptional cases and this is described in a later section.

The scheduler is the part of the supervisor program which determines the order in which the available jobs awaiting execution should be assembled in main store, and should be compiled and executed. In the following sections it will be assumed that no special priority has been attached to jobs by the operator, and that the scheduler is permitted to select jobs in any order; we shall consider later how this order may be influenced by the action of operators.

The advantage of a computer organized scheduling system

After recognizing that the proposed dynamic buffering scheme (described above) on Atlas means that normally all slow peripheral transfers are indirect and overlapped with computing operations, the merit of doing anything other than executing jobs in precisely the same order as they enter the computer might be open to doubt.

However, this simple method of operation can lead to extremely inefficient use of the computer. This would be the case, for example, if a long sequence of jobs all produced output for one peripheral, while other peripherals, which later jobs wished to use, remained idle. On a very large computing system (32 tape decks, 4 line printers, etc.) the order in which jobs are executed is not so important since there may be enough peripherals to deal with any load, irrespective of the order in which jobs are executed.

However, it is uneconomical to buy a computing system with more than the minimum number and types of peripherals, which, if used efficiently, can handle the expected load.

If jobs are sorted by the operator before they enter the computer, inefficiencies of this type are less likely to occur. However, due to the speed at which the machine executes jobs, and also the operator's comparative lack of knowledge of the immediate state of the machine, the task of efficiently ordering a large number of jobs is extremely difficult. A far better method of operation is for the operator to take no account of the types of job entering the computer (all jobs being fed in as soon as they appear), and for a scheduling system, which at any instant knows the exact state of the computer, to determine the order of execution.

A sophisticated and efficient scheduling system is the ultimate object of the overall Atlas supervisor, made possible by the special time-sharing features of the Atlas hardware and basic supervisor routines. If a computing system does not have built-in features for protecting and interrupting programs during execution, then anything other than a very simple scheduling system is probably not worth-while, even if it is possible to design some elegant system. In these circumstances the advantages of an "efficient" scheduling system are offset by the time and effort required to devise the system, and then, when the scheduler is working, by the time spent in the scheduling routines.

On Atlas, the main difficulty lies not so much in the implementation of any given scheduling system, but in formulating the particular rules and framework of the system. If the scheduler is designed properly, it should be possible, in the light of operating experience, to tailor it to obtain the best results for the particular installation concerned and type of work it is doing. Excessive scheduling, when considerable time is spent in routines doing detailed look-ahead processes, should be avoided: inaccurate estimates and error monitoring of jobs tend to nullify any attempt at detailed future planning.

The task of the scheduler

The object of the operating system on Atlas is to maintain the fullest possible useful activity in those parts of the computing system which can function simultaneously; that is, to reduce to a minimum periods of idleness in any part of the system which is required for further use. Such periods can be caused by a delay in passing information from one branch of the system to

another; for example, if a job is using a magnetic tape, either the central computer may be idle awaiting the conclusion of a transfer between store and tape, or the tape may be idle awaiting a command from the central computer. A period of idleness can also occur when a peripheral is free and no attempt is being made to execute available jobs which could use this equipment.

The scheduler may be visualized as arranging for the transfer of jobs from a list of available jobs to a list of jobs in course of execution. The supervisor program on Atlas permits sharing of the central computer between several object programs in course of execution, and thus the Execute List may hold more than one job at any one time, if this is necessary to achieve efficiency; the composition of this list will be described in this section. At first sight, it would appear that maximum efficiency would be obtained if the Execute List comprised one job using each output peripheral and magnetic tape, together with a base-load job to use any available central computer time. Due to the wide difference in the rate of use of information by the peripherals and the central computer, the central computer might be expected to be in use for only a small proportion of the total time on any one problem, and whilst not in use for this problem could be used by other problems. The presence of so many problems on the Execute List is not necessary to achieve efficiency, however, and is to be avoided in the interests of efficient use of store. The operative parts of such problems must be in core store to permit fast switching of control between problems without the need for transfers between core and drum stores; this would imply an excessively large core store, any one part of which is only in use for a small proportion of the total time. Inefficient use of store also results from the fact that, for many problems, the combined core and drum store required for execution considerably exceeds that required for storage of input material, and hence total storage requirements are lessened by reducing the number of problems present at any one time on the Execute List. Since frequent switching of control between object programs is not desirable, the supervisor program is designed to provide rapid switching of control between an object program and supervisor routines, rather than between two object programs.

A straightforward reduction in the length of the Execute List is achieved by using the output well in core and drum store and on the system output tape to collect output information from all object programs for all output peripheral equipments. Problems which are output limited (that is, those which use an output peripheral for a longer time than they use the central computer) can then be executed in series, filling the output well. During their execution, such jobs are not held up for peripheral transfers. Only when the output well is filled for all peripherals is control switched to programs which are computer limited, only one of which need be in course of execution at any one time. The output well is emptied by the output peripherals until a low level of available output is reached for any

peripheral, when output-limited jobs are again executed in series to refill the output well. In the absence of jobs using magnetic tape, therefore, the presence of two jobs on the Execute List achieves efficiency, and for most of the time only one of these is actually active. Jobs using magnetic tapes cannot be dealt with in this way, since it is impractical for the supervisor program to provide an adequate buffer for information transferred to and from magnetic tapes. If tape-limited problems are run, therefore, it is essential in the interests of efficiency to maintain these on the Execute List in addition to the two entries described above; the central computer will use tape waiting time to proceed with other jobs in the Execute List. The number of such jobs will depend upon their computing time, and the number of tape decks required; one such problem may be sufficient under normal conditions on many Atlas installations.

The object of the scheduler is to maintain a supply of problems to the Execute List, and to arrange as far as possible that a vacancy on the Execute List can be filled immediately by a problem already assembled in the input well in main store. The scheduler will therefore be activated whenever a vacancy appears on the Execute List, through termination of a problem, and whenever input of a job through the input peripheral equipments is complete. Since no peripheral-limited problem (i.e. problems which are entered to the Execute List in order to supply output to the peripherals) will be executed once the output well is full, and while the peripheral equipments are emptying it, the scheduler is also activated whenever the level of available output for any peripheral which is in use reaches a lower limit. Problems are then executed in series until the output well is filled, and then a vacancy is left on the Execute List. By this means, frequent interruption of a long computer-limited job by short output-limited jobs is avoided; the latter are run in bursts, during which the computer-limited job becomes inactive, and between these "bursts" the computer-limited job is free of interruption. Following sections will describe in more detail the types of jobs handled by the scheduler, and it will be found that the simple description so far given of the necessary entries in the Execute List is capable of extension to embrace all jobs which occur in practice.

Streams

Having stated the general aims and method of the scheduler there remains the practical problem of selecting the right type of job from those available, and assembling it for execution. The major difficulties connected with selecting jobs of the right type from a single long list of jobs are the time required for searching, and the possibility that some job may be permanently overlooked since it never fits into the desired category. A simple solution to these difficulties is obtained by sorting jobs into streams as they enter the computer. For scheduling purposes there is need for a computer and a tape stream, and for various peripheral streams. The tape and peripheral streams supply jobs which will maintain steady

use of the tape decks and peripheral devices, whilst computer stream jobs keep the central computing unit active.

There are several factors which influence the types of jobs which enter the various streams; these are now listed and their effects on the composition of the streams discussed.

1. The most obvious way of classifying jobs into streams is according to whether they are computer, tape, or peripheral limited.
2. With short jobs where execution time is less than some specified time (say one minute) there is a strong possibility that estimates for the use of peripherals, tapes, and the central computing unit are inaccurate. This is particularly true of "common" jobs which contain no job description of their own but use the standard Atlas one.
3. If long and short jobs occur in the one stream, there is the possibility that a large number of short jobs may accumulate during the execution of a long job, and this is undesirable.
4. Since a jobs' output may occur in uneven bursts, a long job, even if it is peripheral limited, cannot be relied upon to maintain steady use of the peripherals.
5. For reasons already mentioned, peripheral-stream jobs are to be executed in series, and hence they must be short in order to ensure that all the peripherals are regularly supplied with output.
6. Since there are only a finite number of tape decks there is need for a tape allocation routine which governs the allocation of tape decks; this is most easily implemented if all jobs which require tape decks fall into only one or two streams.

Taking all these factors into account, a reasonable solution as to the composition of the stream lists, which fits in with the rest of the scheduling system, appears to be:

1. All short jobs which do not use tapes are allocated to the stream belonging to the peripheral which they use most.
2. All short jobs which use tapes, and all long jobs where tape time exceeds their computing time, are allocated to the tape stream.
3. Remaining jobs, that is all long jobs which are not tape limited, are allocated to the computer stream

Jobs are sorted into their streams as they enter the computer, according to the information given in the job description. When a particular type of job is required on the Execute List, the appropriate stream is consulted and the first job chosen; by this means the search time is reduced to a minimum and there is no possibility of any jobs being overlooked. When a computer or tape job leaves the Execute List it is replaced by a job from the computer or tape stream. When a peripheral job ends and there is a peripheral whose backlog is below a certain desired level, or when there is no peripheral job on the Execute List and the output backlog for any peripheral falls below an emergency low level, then a

job from the relevant peripheral stream is entered on the Execute List. Also, every time a new job joins a stream queue and becomes available for execution, the scheduler is consulted to see if this job is wanted on the Execute List. Once the tapes for a short tape job have been mounted, and it is not entered in the Execute List as a tape job, then it may be treated as a peripheral job and selected for execution on this basis; this affords a means whereby short tape jobs may by-pass long tape jobs. Some peripheral-stream jobs may be willing to have their results put out on any one of a number of peripherals, and jobs in this category are queued in an "any" stream. In this case the scheduler puts out the results on the peripheral which is being used the least, thus helping to spread the output load evenly over as many peripherals as possible.

Since the stream queues must contain entries for every complete job in the computer they may become very long. It is essential, therefore, that entries in these lists be as short as possible to permit of efficient scanning by the scheduler and tape and space allocation routines without excessive use of central computer time, core store, and transfers to and from the drums. The simple method of choosing jobs in order from the stream queues for the Execution List make it possible to implement the system if each entry only contains a condensed form of the job title, some reference position of the job on tape, and a link with the next job in the stream. However, jobs have to pass through an assembly stage (see next section) before entry to the Execute List, and it is convenient if there is sufficient information (i.e. number of tape decks and system tapes required) in the stream queue to tell whether it is possible to go ahead and prepare the job. Also, in certain special circumstances it may be necessary to execute jobs out of their natural stream order. For these reasons approximate estimates of execution time, amount and type of output and space requirements, the type of compiler to be used, and any external priority of the jobs are kept. This information is also valuable to the routines which govern the allocation of tape decks and space. The whole of this information is kept in 120 bits, which means that a single block of store (512 words) is sufficient to list over two hundred jobs which may be present in the computer at any one time. Precise information (names of tapes and system documents, location in store and on tape of each document, etc.) necessary for the assembly and preparation of jobs is contained in an "internal" job description, which is formed at input time from the programmer's job description and is recorded in an input stream as a separate document belonging to this job.

The question arises as to the action taken when a peripheral or tape becomes free and there are no jobs awaiting execution in that particular stream. Under the present stream structure it is possible that there are jobs in other streams which may make a great deal of use of this peripheral or tape deck; this is the case, for instance, with long peripheral-limited jobs which have been placed in the computer stream. When a stream becomes empty

it therefore appears worth-while to search and see if there are jobs in other streams which could profitably be transferred to the empty stream. However, the fact that the stream is empty means that this peripheral is not likely to be in great demand in the near future, and hence there is no point in transferring jobs, which make little use of the free peripheral, to the empty stream; in fact doing this could lead to inefficient operation. In order to ensure enough work for the central processor there should always be at least two jobs on the Execute List, even if it means taking jobs from the same stream.

Assembly of jobs

The computing speed of Atlas is such that small problems may be expected to occupy the central computer for only a few seconds during their execution. It is therefore essential that an entry in the Execute List should be able to be replaced rapidly by the next problem selected by the scheduler, and one of the tasks of the scheduler is to assemble problems in advance so that compiling and execution may begin immediately and are not subject to delays. The scheduler in fact selects in advance problems which are to be assembled, deals with any long-term preparations and then transfers these from the list of available jobs to an Active List, which comprises jobs in the course of assembly prior to execution. Normally computer and tape jobs are not entered on the Active List till the previous computer and tape jobs are finished; since these jobs are long it is difficult to predict when the previous job is likely to finish, and a short delay between the running of these jobs can be tolerated. Peripheral jobs are entered on the Active List when they are likely to be selected for execution, and where possible a certain minimum of jobs are kept on the list to ensure a ready supply of prepared jobs for the Execute List. When a problem is required for execution, only those problems on the Active List whose assembly is complete are considered, with the exception of when the central computer can only be used to start execution of a partially assembled problem.

Assembly of a problem involves collecting all the information required to run a problem so that it is available to the central computer with the minimum possible delay when the problem is entered to the Execute List. A completely assembled problem on the Active List has the relevant input information in the input well in the combined core and drum store, the required compiler or input routine in the drum store, and any private magnetic tapes mounted and their titles checked. The process of assembly may therefore consist of reading input information from the system input tape, reading documents previously recorded on previous system tapes ("document" or "archive" tapes), reading a compiler into store where necessary, instructing the operator to mount magnetic tapes, and verifying the titles of all such tapes. The system may be easily extended to permit "on-line" use of other peripheral equipments which would be treated as being in the same class as magnetic tapes and prepared for use in a similar manner.

It is essential to observe that assembly of a problem may extend over a long period of time, although, of course, the central computer is only occupied for a small proportion of the time. Collection of a document from the system input tape may not be completed until a full "swing" of this tape has elapsed, a time interval, in the worst case, of around 16 seconds. Since the same tape is used both to write blocks received from input peripheral devices and read back previously written blocks to recover documents as they are required, the tape performs a "swing" action in which frequent scans are made over a few feet of tape, although it will gradually progress forwards. Collection of documents from other system tapes may require scanning over long sections of tape, and a delay of up to 5 minutes may be experienced even when the correct tape is already mounted. The mounting of magnetic tapes by the operator may be expected to occupy a variable length of time, but one which is necessarily long considering the speed of the central computer. The logic of the scheduler and the assembly routine takes into account these wide variations in assembly times, and attempts to minimize the effect of delay in any branch of the system on the other branches of the system which can operate concurrently.

The process of assembly is divided into two main phases. The first phase deals with long-term assembly of magnetic tapes, documents from system tapes other than the system input tape, and compilers. Not until this phase is completed is a job made available to the scheduler for inclusion on the Active List, and not until called for by the scheduler is the assembly of any documents from the system input tape initiated. The space and tape allocation routines, to be described later, scan the complete job list and initiate the first assembly phase for jobs near the head of the stream queue. As tape mechanisms become available, the operator is instructed to mount private tapes and/or system tapes; the required documents are read from system tapes into main store and from there may be temporarily recorded on the system dump tape. The space allocation routine makes main store available for compilers, which, if they are not already in main store, are read as required from a system library tape. As this stage of assembly of each problem is completed, the scheduler is activated, and should the problem be required on the Active List, any relevant documents are collected from the system input tape. The problem is, in fact, entered to the Active List, and is scanned by the routine controlling the system input tape; this routine collects documents required by any job on the Active List as the system input tape moves forwards to the writing position, and hence can assemble several problems during one "swing" of the system input tape. The location of documents is obtained from the internal job description which, if necessary, is read in from the system input tape on the backward swing. It should be noted that any or all of the phases of assembly are omitted where they are not required. Frequently a problem will require only information recently read in through the input peripherals

and will require a compiler already present in main store, in which case the first assembly phase is omitted. If demands on store are not heavy, the problem may also be already in main store, in addition to being recorded on the system input tape, and in this case, assembly is virtually instantaneous.

Interaction with operators

The scheduler can be independent of any operator scheduling, but there does exist a two-way communication system between the operator and scheduler. When the scheduler requires action by the operator, such as the mounting of magnetic tapes, an explicit command is printed out to the operator. Because of the comparative slowness of operator action, requests to the operator are given, where possible, well in advance of when the action is needed. In fact, it is advisable that jobs which require some operator action should be fully prepared before entering the execution phase, otherwise inefficiencies resulting from a prolonged delay in operator action can be serious.

It is possible for the operator to convey information to the scheduler, and an example of this is the allocation of an external job priority. From the scheduler's viewpoint external priorities given to a job through the operator at the user's request are necessary to satisfy the user, but may lead to gross inefficiencies in computer operation. The need to obtain the operator's approval before allocating a job special priority affords some protection, but it should be recognized that external priorities impinge on the scheduling system and determine independently what is to happen. Of course the change of state in the computer brought about by the allocation of an external priority is taken into account by the scheduler when determining the best course of action in the future. Also, the framework of routines set up to implement external priorities may be used by the scheduler to achieve its own ends.

The normal scheduling system on Atlas is based almost solely on efficiency considerations, and allocation of an external priority to a job by the operator affords the means whereby the individual user's requirements may be satisfied. There appears to be a need to allow four priorities—top, high, normal, and low, with the following functions.

1. Top priority: the specified job is executed and the results put out as soon as possible regardless of the state of the computer, or what other jobs are awaiting execution.
2. High priority: the specified job jumps to the head of its stream queue.
3. Normal priority: the specified job is given the priority and treatment described in previous sections.
4. Low priority: the specified job is treated as normal until it reaches the execution list when it remains at the bottom of the list.

Tape allocation routine

The Atlas operating system normally requires three tape decks for its own use, viz. input tape, output tape, and dump tape; this means that three decks are occupied permanently by the supervisor, though the system is flexible enough to operate with fewer decks when absolutely necessary. When only two decks are available the input and output tapes are combined on one tape. Both tapes are normally used in a similar fashion, and the cost of combining them into one is a reduction in the effective size of input and output wells on tape. If only one deck is available this must be used by the dump tape, which acts as an extension of main store. The operating system still functions when no decks are available, but only as efficiently as circumstances will allow, and care must be taken that input and output wells are not allowed to occupy all the free space in main store. If a computer installation has a large number of decks then it may be worth while for the operating system to use more than three decks; for instance, it would be valuable to have one deck permanently loaded with the library tape and another with a separate dump tape to record error monitor dumps of object programs.

Part of the information which accompanies each job into the computer is a list of the tapes required and any documents which have to be obtained from system tapes. Thus, before a job commences execution, the number of decks it requires and the names of the tapes to be mounted are known; this information is valuable for scheduling purposes and the assembly of jobs prior to execution. However, in certain cases tapes are only required during part of their total time of execution, and thus tape decks can be called for or made free during the course of a job's execution; one common example is the use of tapes as temporary working space during compilation.

It is the job of the tape allocation routine to allot tapes to the various sources which require them in such a way that the scheduler can maintain efficient operation of the computer. There is clearly strong inter-action between the scheduler and the tape allocation routine, the behaviour of one affecting the decisions of the other. The normal order of priority for allocating tape decks is as follows:

1. Supervisor system tapes.
2. Jobs or compilers which call for tapes during execution.
3. Jobs called to enter the execution list and requiring either private tapes or documents from system tapes.
4. Jobs near the head of their stream queues which require private tapes or documents from system tapes.

Since jobs have not direct access to system tapes and it is the supervisor's job to collect documents from these tapes, they can be dismounted as soon as the relevant documents are read into main store. However, there is no point in dismounting a system tape if the deck is not required, and if the deck is required it is worth-while,

before dismounting the tape, to read off any other documents which are likely to be called for in the near future. Exactly how many documents should be read into main store depends on the size of the documents, the state of the dump tape, and the present demand on tape decks. If the space occupied by documents which are read into main store before they are called for is wanted, then the documents are recorded on the dump tape. It should be noted that apart from this reason the dump tape is used to record error monitor dumps, suspended programs, and overflow from main store, which could include such items as jobs in process of execution and documents or output which lie outside the scannable region of the output tape.

Though it is not customary to use a satellite computer in association with Atlas operation it is possible to do so if required. Jobs previously recorded on a "satellite" magnetic tape are executed in sequence, and the outputs from these jobs are recorded on another tape to be printed on the satellite computer. Tapes in this category are mounted as soon as sufficient tape decks are available after the request for a satellite run has been received from the operator.

Space allocation routine

Every block of main store space in Atlas is a member of a group, depending on its owner, and there are six main groups of owners; these are:

- (a) Free space.
- (b) Input and output wells.
- (c) Jobs that are being prepared for execution.
- (d) Jobs being executed.
- (e) Compilers.
- (f) The supervisor.

The basis of the space allocation routine is that a hierarchy is formed of all these owners and also of all requests for space, and no request is allowed to take space from an owner of higher priority. The ratings of each member of this hierarchy are essentially dynamic, depending on the state of the machine at the time.

There is an area of space in the machine which is readily available to any request; this consists of all free space and all blocks in input and output wells which have been duplicated on magnetic tape, copies of which have been kept in main store. The number of these available blocks is known at any time. Essentially, a request can only take space from this group, and if there is not sufficient space, then routines are activated which begin to free blocks of a lower priority than the request; when this happens, the request is put into the space request queue, and each time a block is freed, the top element in the queue is checked to see if there are enough free blocks for it. In certain cases the request is not queued but the space allocation routine returns to the routine which made the request with the information that the request has been refused, thus allowing the routine to take alternative action. For instance, the scheduler may ask for space to enter a peripheral job in

the Execute List; if insufficient space is available, it may be possible for another smaller job on the Active List to fit into the space available. Certain other requests are allowed to by-pass this main queue; these are the types of requests which ask for single blocks at relatively infrequent intervals, and if the request is not granted some part of the computing system would be halted; i.e. if a request for input-well space is not granted then a peripheral may be halted.

The ordering of the space request queue is important because low priority requests for a large amount of space should not be allowed to block small, relatively high priority requests. Of course, care must be taken that a request is not always by-passed, and if it has been in the queue for more than a certain length of time it is allowed to begin to reserve space. Also, the priority of most requests will be proportional to the length of time that they have been in the queue.

There are two types of information in main store. The first is information which has no copy on magnetic tape, i.e. programs being executed. In order to free this type of space it is necessary to dump the information on the dump tape. This can take a long time and owners in this class have a fairly high rating. However, the second class is information which is duplicated on magnetic tape, i.e. compilers; space belonging to these owners is overwritten and freed almost instantaneously.

It is a complicated task to keep track of the information in the store and on the dump tape, but a directory system has been devised which makes it possible to overwrite peripheral-stream information in units of one block. However, for other information it is necessary to dump or overwrite it in reasonably sized sections; i.e. even if only a few blocks are needed an entire document might be overwritten.

The actual routines which do this work are in two sections. One section is in the fixed store, and this is sufficient to take care of most requests for space. The other section is in the main store, and this contains the longer routines which deal with dumping and retrieving information. These space allocation routines, and in fact the whole supervisor, have been written in such a manner that it is possible to overwrite the major part of the main store routine, leaving virtually the entire store available to the user.

Conclusion

It must be emphasized that the system outlined above is suitable for any type of Atlas installation, and is independent of the configuration of peripherals, core store, and drum store, apart, of course, from changes in essential parameters at different installations. However, the scheduling system has been designed in two parts. Routines called into action most frequently are held in the fixed store, and these will be the same on all installations. They are called into action, and effectively controlled, by routines in the core and drum store and by parameters in the subsidiary working store, which can be changed in the light of experience, and to meet any particular requirements.

Acknowledgements

This work forms part of the Atlas project. It has benefited from many helpful discussions with the authors' colleagues at Manchester University and Ferranti Ltd., whose permission to publish is acknowledged.

References

1. KILBURN, T., HOWARTH, D. J., PAYNE, R. B., and SUMNER, F. H. (1961). "The Manchester University Atlas Operating System Part I: Internal Organization," *The Computer Journal*, Vol. 4, p. 222.
2. HOWARTH, D. J., PAYNE, R. B., and SUMNER, F. H. (1961). "The Manchester University Atlas Operating System Part II: User's Description," *The Computer Journal*, Vol. 4, p. 226.
3. KILBURN, T., PAYNE, R. B., and HOWARTH, D. J. "The Atlas Supervisor," *Proc. E.J.C.C.*, December 1961.

Editorial Note:

A Summary of the above paper was presented at the **Interdata Exhibition** in Munich, Germany, on 30 August 1962 by P. D. Jones, with parallel French and German translation. The German version of this paper appeared in *Elektronische Rechenanlagen*, Heft 4, 1962 (Oldenbourg Verlag, Munich), and a French version has been submitted to *Chiffres* (AFCALTI, Paris). The original text, as above in English, was received by the honorary editors of this *Journal* on 21 June 1962. Overseas readers may find these references of interest.