bcc	title Basic System Resource Allocation	prefix/class-number.revision BSRA/W-27
checked checked Larry L Barnes approved Mel	authors Butler W. Lampson	approval date: 9/10/69 revision date: classification: Working Paper distribution: Company Private pages: 11

ABSTRACT and CONTENTS

The basic system mechanisms for allocating resources and enforcing restrictions on the use of resources are described. A distinction is made between allocation of disk space and all the other resources.

Introduction

From the viewpoint of the basic system, accounting and resource allocation are completely separate activities. The system provides facilities for distributing resources to processes. It also keeps track of resources actually expended, which may be much less than the resources which were distributed. Decisions about how to convert this information into charges, however, are beyond the province of the basic system.

This document is concerned with the basic system facilities for distributing resources. As elsewhere in the system, the philosophy is to provide as little machinery as possible, and in the most general form. Considerable effort is therefore required to convert the facilities described here into a convenient marketplace for resources. It is believed, however, that this conversion is possible. Presumably it will be done by the utility.

An important principle underlying the resource allocation mechanisms is that it must be possible to allocate resources for a limited period of time, after which they become unavailable. For certain resources, such as CPU time, this can easily be done by attaching an expiration date to the resource allocation, which thus might take the form

2% of 1 CPU from 1000 to 1200, July 29, 1969.

Possession of such a resource allocation is of no value after the expiration time, since the system will not allow any computing to be charged against it.

Unfortunately, there are resources for which an expiration time is not a satisfactory mechanism for control. Disk space in particular falls into this category, for two reasons. First, it is impractical to keep track of the expiration time for every disk page ever assigned, especially when this time may frequently be extended. Secondly, all disk space would not have the same expiration time, and it is not at all clear how a reasonable decision could be made about which allocation to use each time a page is required.

An alternative strategy for limiting such non-expiring resources as disk space is to make sure that they are recoverable, i.e. that the original owner who passes them on to a client can always take them back. This approach has several implications:

recoverable resources cannot be passed freely from one user to another. Instead, their transfer must remain under the control of the original owner so that he can keep track of their location;

the owner must have the authority to take back the resources he has given out. The client thus remains at the mercy of the owner;

the owner must have authority to destroy objects (such as files) in order to release resources so that they can be taken back;

recoverable resources from different owners may not be mixed, either for bookkeeping purposes or to obtain the

necessary resources for the creation of a single object. For each object using resources, there must be a way of finding the owner.

The only recoverable resource in the system at present is disk space. Its treatment is discussed in detail below.

Resource Allocations

A new kind of object is provided to keep track of the ownership of expiring resources. It is called a resource allocation (RA) and is kept in an MIB exactly like a file, process or access key. It resembles the latter in that it is entirely contained in the MIB.

An RA consists of a validity period and a resource list. The latter is the same for all RAs and comprises

Number of teletype lines
Number of processes
Guaranteed minimum CPU share
Maximum CPU share
Guaranteed minimum number of drum pages
Maximum number of drum pages
Guaranteed minimum number of disk accesses/ 30 seconds
Maximum number of disk accesses/ 30 seconds

Each field occupies a half-word (12 bits). The whole requires 4 words. It is convenient to denote the resource list by r_i.

The validity period contains four 24-bit numbers M_i which are times measured in minutes since the origin of the clock. Positive numbers suffice to count for 15 years. The two kinds of RA are differentiated by their validity periods:

a fixed RA (FRA) has a starting time $st=M_1$ and an ending time $et=M_2$. For a FRA, $M_3=-1$ and M_4 is ignored. The meaning is that the resource list is valid (i.e.. those resources are available) for the period (st,et). If st = 1200, et = 1300, then the resources are available for exactly 100 minutes. Notation for a FRA is

$$(st,et:r_i)$$

a repetitive RA (RRA) has a starting time $st=M_1$, an ending time $st=M_2$ an interval $i=M_3$ and a repetition period $p=M_4$. The meaning is that the resource list is valid for time periods $(st+j*p,\ st+j*p+i)$, as long as $st+j*p+i \leq et$. Thus if st is 9:00 a.m., July 28, 1969. et is midnight December 31, 1969, i is 1 hour and p is 24 hours, the resources are available every day from 9:00 a.m. to 10:00 a.m., starting on July 28, 1969 and ending on December 31, 1969. Notation for a RRA is

$$(st,et,i,p:r_i)$$

Operations

The following operations on RAs are possible. Ri denotes the information required to define an RA. It is the same as the information required to define a file to the basic system.

READ'RA (R,Array A) reads the items of the RA into the array. Requires read access. Works on all PRs.

SPLIT'RRA (R1,R2,R3) requires R1 to be an RRA, R2 and R3 to be null RAs. Sets

$$R2 \leftarrow (st^1, \min(st^1+i^1, et^1))$$

$$R3 \leftarrow (st^1+p^1, et^1, i^1, p^1 : r_i^1)$$

I.e., splits off the first valid period defined by R1 into R2 and leaves the rest in R3.

The next four operations require all the inputs to be FRAs, all the outputs to be null RAs.

SPLIT'TIME (RA1, RA2, RA3, S). Sets

$$RA2 \leftarrow (st^1, S : r_i^1)$$

$$RA3 \leftarrow (S, et^1 : r_i^1)$$

Requires $st^1 \leq S \leq et^1$. This operation splits RA1 into two new RAs at the time given by S. It thus allows subdivision of resources by time.

MERGE'TIME (RA1, RA2, RA3). Sets

$$RA3 = (st^1, et^2 : r_i^1).$$

Requires $et^1 = st^2$, all $r_i^1 = r_i^2$. This inverts SPLIT'TIME

SPLIT'RES (RA1, RA2, RA3, ARRAY S). Sets

$$RA2 = (st^1, et^1 : S[i])$$

$$RA3 = (st^1, et^1 : r_i^1 - S[i])$$

Requires all $r_i^1 - S[i] \geq \emptyset$ and S to have at least 8 elements. The array S defines a division of RA1's resources between the two outputs.

MERGE'RES (RA1, RA2, RA3). Sets

$$RA3 = (st^1, et^1 : r_i^1 + r_i^2)$$

Requires $st^1 = et^1$, $st^2 = et^2$. Inverts SPLIT'RES.

All operations permit the same RA to be used for input and output because all are implemented with the following algorithm:

- read data from all input RAs and make them null then write results into output RAs.

Resource allocation and processes

Every process has a current resource allocation (CRA) which contains exactly the same information as a FRA. A process may be specified in place of a FRA in any of the above operations, and its CRA will be referenced. The system uses the CRA of a process to determine what resources it has. In particular, the number of processes must be at least one, since the existence of the process itself (actually its activity or presence in PRT) is charged against this number, and the number of lines must be at least one if the process has a line. When the et of the CRA is reached, the system looks in the MIB of the owner of the process for an FRA called <process name>:NRES and copies it into the process CRA if it is found. If its st is > the current time, or if no such FRA exists, the process becomes dormant. If the supply of drum pages is decreased, the process' APS is reduced in size. If the number of processes is <1, the process becomes dormant. If the number of lines is <1 and the process has a line, it is released.

It is assumed that if the process is expected to run beyond the et of its CRA, someone, most likely the utility system running in that process, will set up < >:NRES in time. A plausible strategy for doing this is to arm a real-time interrupt to expire a few minutes before et (CRA). When the interrupt occurs, the utility can obtain the next increment of resources from its own RAs.

The process creation and destruction operations (see PROCD/W-) provide for the initial assignments of resources and for returning leftover resources to a FRA.

Disk Space

The treatment of disk space is quite different from that of other resources for two reasons:

it is a recoverable resource (see introduction) and lacks an expiration time

it is assigned to MIBs rather than to processes.

The scheme described here is somewhat less general than it might be. It has been chosen because it seems to be adequate, and in order to avoid complications.

In the header of each MIB is a word called the disk allocation (DA) which specifies the number of pages of disk storage which may be allocated to files in that MIB or to private pages of processes in that MIB. Each time a page is allocated the count is decreased by 1, and each time a page is released it is increased by 1. If it is Ø and an attempt is made to allocate a page, the attempt fails for lack of space.

For each MIB there is also a flag called the frozen disk allocation flag (FDAF). The first entry on the owner access lock list is called the patron. Transfers of disk space from one MIB to another are performed by the following operation

```
     MOVE'DA(MIB1,MIB2,INTEGER N);
```

The MIBi are (user number, disk address) specifications of two MIBs. If the operation satisfies certain restrictions, N is subtracted from DA(MIB1) and added to DA(MIB2). It fails unless

1) DA(MIB1) - N $\geq$ Ø

2) Either

a) FDAF(MIB1) is clear, the caller (i.e. the current UNO) has owner access to MIB1 and is the patron of MIB2, or

b) FDAF(MIB1) is set, the caller is the patron of MIB2 and has owner access to MIB1.

The first case corresponds to transferring disk space from a group to a user, the second to taking it back. Normally the user's disk space will be frozen, so that it cannot move anywhere except back to the patron, and the group's space will be unfrozen. Normally the system owner will be patron for a group.

This machinery gives no direct help in allocating different classes of disk space, e.g. permanent and scratch. The basic system does provide a two-bit field, called the file durability, with each file. It places no interpretation on this field. The intention is that the group utility will take the value of the field as an indication of the permanence of the file and will delete files with small durabilities first if forced to take such measures.

Suppose then that a group wishes to sell a user 200K of disk space for 10 minutes. It makes a MOVE'DA call to transfer the 200K to the user. Then 10 minutes later it attempts to take it back. If the user's DA is less than 200K, it scans his MIB and deletes the contents of files, starting with those of low durability, until 200K becomes available in his DA.