Note: This was originally found at http://www.isham-research.com/zarking.html

It was periodically re-published in the University of Michigan Computing Center (later called the Information Technology Department) Newsletter, although I recall it being attributed to someone (probably Jeff Berryman) at the University of British Columbia

The Paging Game

by Jeff Berryman, University of British Columbia, 1972

The Paging Game was written by Jeff Berryman when he was working on project MAC, specifically one of the virtual memory systems. The version best known to IBM and MVS sites from the mid-1970s onwards is first, followed by the original:

THE PAGING GAME

*** RULES ***

1. Each player gets several million *things*.
2. Things are kept in *crates* that hold 4096 things each. Things in the same crate are called *crate-mates*.
3. Crates are stored either in the *workshop* or in the *warehouse*. The workshop is almost always too small to hold all the crates.
4. There is only one workshop, but there may be several warehouses. Everybody shares them.
5. Each thing has its own *thing number*.
6. What you do with a thing is to *zark* it. Everybody takes turns zarking.
7. You can only zark your own things, not anybody else's.
8. Things may only be zarked when they are in the workshop.
9. Only the *Thing King* knows whether a thing is in the workshop or in a warehouse.
10. The longer a thing goes without being zarked, the *grubbier* it is said to become.
11. The way to get things is to ask the Thing King. He only gives out things in crates. This is to keep royal overhead down.
12. The way to zark a thing is to give its thing number. If you give the number of a thing that happens to be in the workshop, it gets zarked right away. If it is in a warehouse, the Thing King moves the crate containing your thing into the workshop. If there is no room in the workshop, he first finds the grubbiest crate in the workshop, whether it be yours or somebody else's, and packs it off with all its crate-mates to a warehouse. In its place he puts the crate containing your thing. Your thing gets zarked and you never even know that it wasn't in the workshop all along.

13. Each player's stock of things has the same numbers as everybody else's. The Thing King always knows who owns what thing and whose turn it is, so you can't ever accidentally zark somebody else's thing even if it has the same number as one of yours.

*** NOTES ***

1. Traditionally, the Thing King sits at a large, segmented table and is attended by pages (the so-called "table pages") whose jobs it is to help the king remember where all the things are and to whom they belong.
2. One consequence of Rule 13 is that everybody's thing numbers will be similar from game to game, regardless of the number of players.
3. The Thing King has a few things of his own, some of which move back and forth between workshop and warehouse just like anybody else's, but some of which are just too heavy to move out of the workshop.
4. With the given set of rules, oft-zarked things tend to get kept mostly in the workshop, while little-zarked things stay mostly in a warehouse.
5. Sometimes even the warehouses get full. The Thing King then has to start piling things on the dump out back. This makes the game slow because it takes a long time to get things off the dump when they are needed in the workshop. A forthcoming change in the rules will allow the Thing King to select the grubbiest things in the warehouses and send them to dump in his spare time, thus keeping the warehouses from getting too full. This means that the most infrequently-zarked things will end up in the dump so the Thing King won't have to get things from the dump so often. This should speed up the game when there are lots of users and the warehouses are getting full.

Long Live the Thing King!

And here is something close to the original version, from the Project MAC manual.

THE PAGING GAME

Project MAC Computer Systems Research Division

*** RULES ***

1. Each process gets several million *bytes*.
2. Bytes are kept in *pages* that hold 4096 bytes each. Bytes on the same page have

locality of reference.

3. Pages are stored either in *memory* or on the *disk*. The memory is almost always too small to hold all the pages.
4. There is only one memory, but there may be several disks. Everybody shares them.
5. Each byte has its own *virtual address*.
6. What you do with a byte is to *reference* it. Everybody takes turns referencing.
7. You can only reference your own bytes, not anybody else's.
8. Bytes may only be referenced when they are in the memory.
9. Only the *VM Manager* knows whether a byte is in the memory or on a disk.
10. The longer a byte goes without being referenced, the *older* it is said to become.
11. The way to get bytes is to ask the VM Manager. He only gives out bytes in pages. This is to keep overhead down.
12. The way to reference a byte is to give its virtual address. If you give the address of a byte that happens to be in the memory, it gets referenced right away. If it is on a disk, the VM Manager moves the page containing your byte into the memory. If there is no room in the memory, he first finds the oldest page in the memory, whether it be yours or somebody else's, and packs it off with the rest of the page to a disk. In its place he puts the page containing your byte. Your byte gets referenced and you never even know that it wasn't in the memory all along.
13. Each process's stock of bytes has the same virtual addresses as everybody else's. The VM Manager always knows who owns what byte and whose turn it is, so you can't ever accidentally reference somebody else's byte even if it has the same virtual address as one of yours.

*** NOTES ***

1. Traditionally, the VM Manager uses a large, segmented table and "page tables" whose jobs it is to help the king remember where all the bytes are and to whom they belong.
2. One consequence of Rule 13 is that everybody's virtual address will be similar from game to game, regardless of the number of processs.
3. The VM Manager has a few bytes of his own, some of which move back and forth between memory and disk just like anybody else's, but some of which are just too heavyily used to move out of the memory.
4. With the given set of rules, oft-referenced bytes tend to get kept mostly in the memory, while little-referenced bytes stay mostly on a disk.
5. Sometimes even the disks get full. The VM Manager then has to start piling bytes on the dump out back. This makes the computer slow because it takes a long time to get bytes off the dump when they are needed in the memory. A forthcoming change in the rules will allow the VM Manager to select the grubbiest bytes in the disks and send them to dump in his spare time, thus keeping the disks from getting too full. This means that the most infrequently-referenceed bytes will end up in the dump so

the VM Manager won't have to get bytes from the dump so often. This should speed up the game when there are a lot of processs and the disks are getting full.

Long Live the VM Manager!