**WORKING NOTE AC75**

Alan Creak
4 December 1989

# OPERATING SYSTEMS COURSE STRUCTURE

RATIONALE.

The traditional course in operating systems, as reflected in a number of textbooks, is organised on bottom-up lines. In effect, the treatment elaborates the position, "We have a computer. How can we make it work ?". I have for some time now been less than happy about the bottom-upness of this approach, not for doctrinaire reasons but because I find it increasingly difficult to fit in new material in plausible ways. ( Consider, for example, a file on a Macintosh system : it has two "branches". Unless you begin with an appropriate bottom, up just doesn't lead you there. ) To make it work at all, you have to take as your primaeval computer something which will work as an example of the topics to be included in the course - which fairly constrains you to choose something not too different from the standard time-sharing computer system which was the mainstay of university computing a decade or two ago. There are still lots of such systems around, but nowadays hardly any of the students have met one when they begin the operating systems course. What they have met are microcomputers of all shapes and sizes, WIMPS, modems, bulletin boards, little robots, and so on. It seems obtuse to begin with a strange model when a familiar one is closer to hand.

So suppose we switch our initial model to a microcomputer. Which microcomputer ? At Auckland University Computer Science department, the obvious choice is a Macintosh : but that's, if anything, a more complicated starting point than the old time-sharing system. And what happens in five years' time when fashions have changed again ?

In such a fluid world, how can we devise a comparatively stable course ? We need a skeleton which won't change much as hardware and software developments come and go, but which will nevertheless be sufficiently flexible to cater for all the things we might want to do with our operating systems. It seems to me that the only way to achieve such a desirable end is to change the starting position of the course to somewhere essentially immovable. Is there such a position ? After much thought, I believe that there is, and it is to be found in the demands of the people who use the computer system. I promulgate this principle :

People are the only invariant.

How do we use this invariant ( assuming it really is one ) ? We switch attitudes. Instead of the bottom-up approach I described at the beginning of this note, we start at the top with the question "What do we want computer systems to do for us ?". Seeing that all operating systems are designed to make computers work for people ( despite appearances ), analysing that question must in principle lead us somehow to any material at all which can be classified under the heading of operating systems. In particular, the Macintosh file was designed that way to implement, to some degree, an object- oriented system; and that was considered to be a good idea because it seemed to offer an easier way for people to use computers. And so on.

There is a difficulty with a top-down approach : you can't do it unless you know where you're going. You must have some idea of what's at the bottom if the top-down method is to be sensibly directed at each stage of analysis. This is not a difficulty, as operating systems is offered as a comparatively advanced course, taken by students already familiar with details of computing. Just to make sure the context is there, I think it appropriate to begin with a historical introduction - which is interesting and important in its own right, and also incidentally shows something of how the accent in commercial systems has moved from an obsession with getting the last possible ounce of effort out of the processor to making computers easier for people to use. That conveniently gives a natural lead into the rather unfamiliar treatment in the body of the course.

COURSE.

There follows a suggested course outline, derived by taking the current course topics and writing them in the reverse order. Well, that's not in fact quite true, but the result would have been much the same.

I have started with the question, "What is an operating system ?", and a brief review of the development of operating systems to fill in the background ( to provide the bottom, in the sense used earlier ). With the background established, I can then begin the top-down analysis in earnest. The topics listed are in many cases the same as those which make up the present course; they are, as already observed, in roughly the opposite order. This version is intended only as a sketch, and further analysis may suggest that different arrangements have advantages.

- What is an operating system ?
  - Software to drive a computer.
  - Software to provide service to people.

- Development of operating systems.
  - Early computers.
  - Optimal use of the machinery.
    - Where the waste is.
    - Monitor systems.
    - Multiprogramming.
    - Batch systems.
  - Optimal use of the people.
    - Timesharing.
  - Microcomputers.
    - The system metaphor.
    - WIMPS.
  - The future ?

- What do we need from a computer system ?
  - How do we use computer systems ?
    - Batch and transaction patterns.
  - System management.
  - Communication : instructions, information, assistance.
  - Data management : store, move, and change data.
  - Work : execute programmes.

- System management.
  - Isolated and shared systems.
  - Sharing machines or networks.
  - Managing resources.
  - Managing money.
  - Managing people.

- Communication.
  - Interface with the system.
  - Instructions.
    - Job control.
    - WIMPS.
  - Information.
    - Documentation.
    - Machine-readable material.
      - What's available.
      - What's new.
      - Mail.
  - Assistance.
    - Help.
    - Error identification.

- Data management.
  - Active data : memory management.
    - Virtual memory.
  - Passive data : disc management.
    - What the disc is used for.
      - File systems.
      - Temporary data : spooling, messages, etc.
      - Virtual memory.
    - Disc allocation.
  - Data in motion : input, output, and communications.
    - Streams.
  - Security and protection.
    - Protection.
      - Duplication : file generations, and other methods.
      - Backup.
      - Archiving.
    - Security.
      - Passwords and capabilities.
      - Protection codes and access lists.
      - Supervisor calls and protection rings.

- Executing programmes.
  - Processes and processors.
    - Process table.
  - Memory allocation.
    - Using memory.
    - Using virtual memory.
  - Processor allocation : scheduling.
    - Batch and timesharing systems.
    - Process state transitions.
    - Deadlock.
  - Using devices.
    - Using general devices.
      - Device independence.
    - Using disc files.
    - Using terminals.
  - Housekeeping.
    - Command files.
    - Transaction logs.
    - Event logs.
    - Starting and stopping.