The Edinburgh Multi Access System (EMAS)

Bob Eager FBCS

BCS Kent Branch

6th June 2024

http://emas.bobeager.uk

Introduction

- Some basic information about the Edinburgh Multi-Access System (EMAS)

  - **What?**
    - An efficient multi-access system offering interactive and batch facilities on ICL and IBM (+clone) mainframes
  - **When?**
    - From 1970 to 1992 (predecessor/development between 1966 and 1970)
  - **Where?**
    - Mostly the University of Edinburgh, also the University of Kent
  - **Who?**
    - Developed at University of Edinburgh by Department of Computer Science and Edinburgh Regional Computing Centre; Kent work by Bob Eager

Part I - History

- 1966 saw the foundation of the Edinburgh Regional Computing Centre; in December, a rented KDF9 was delivered to provide a service; significantly, it supported Atlas Autocode
- In the same year, a large team (Edinburgh and ICL) started the Edinburgh Multi Access Project, to write a multi access system for the English Electric Leo Marconi System 4/75 (the most powerful system in their range); the team was led by Harry Whitfield but was really an ICL model
- It included managers, team leaders, designers, 25 programmers, also technical and coordinating committees
- A language called IMP was developed as an implementation language
  - Very similar to Atlas Autocode
- The System 4/75 was delivered in December 1968/June 1969, and the KDF9 was removed shortly afterwards; only a batch service was offered
- EMAP development did not go well, and the project was abandoned in September 1970, as it did not meet expectations

- This was the start of the internal EMAS project, salvaging some of the work, with a team of between 3 and 7 people
- A pilot EMAS service was running within a year, and a formal EMAS service started in October 1972, although it quickly became overloaded
- Replacement with an IBM system was not permitted, so it was to be one of the New Range machines from ICL (a 2980), and it had to be running VME/B
- The 2980 was delayed by a year, so ICL provided a temporary system; this was an ICL 2970 (about 30% of the power of a 2980), with a smaller memory (1MB); running VME/B, this supported a batch system and 1-2 (!) interactive users
- At this point, the EMAS team looked at the feasibility of implementing EMAS on the ICL 2970
  - … but with network communications instead of large multiplexers
  - It would use a PDP-11 as a front end to the rest of the growing network
- The 2900 series was a good architecture for high level languages such as IMP, in which the system would naturally be implemented (IBM architecture was not built for stacks)

- They did it! There are papers about it, notably one entitled:

  An Experiment In Doing It Again, But Very Well This Time

  - The system was called EMAS 2900, but later was also known as **EMAS-2**
- A pilot service worked well, and supported 30 terminals on a 1MB machine
  - ICL donated the 2970 to ERCC
- A full service was offered on the 2970 from October 1978

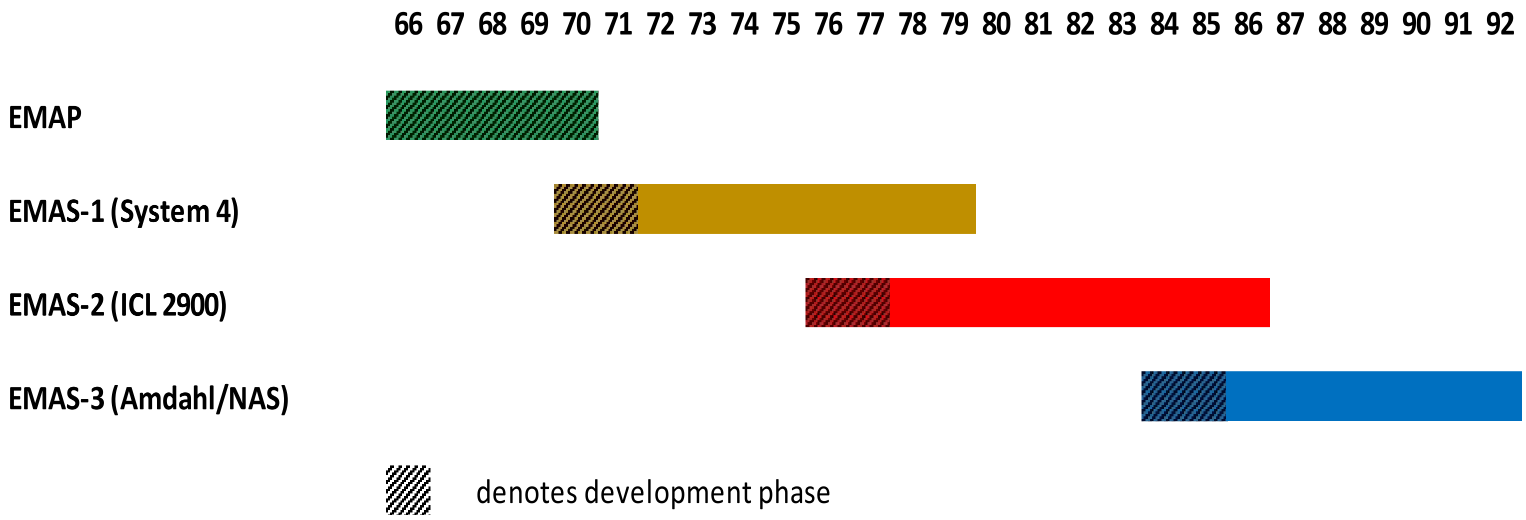
- Meanwhile, in April 1978, the University of Kent installed a copy of EMAS in order to perform trials (see later)
  - Their 2960 was roughly half the power of the 2970, but had 2MB of memory

- By October 1978, the 2980 (running VME/B) had failed to meet its benchmark, although reliability had improved (not perfect)
  - The system was then adopted for regional service
  - A year later, it was clear that the 2980 service was inadequate
  - It was decided to switch it to EMAS, and this happened in January 1980 (two weeks after Kent had done the same)
- Over the next few years there were various upgrades from ICL

- In 1984, an Amdahl 470/V7 was delivered
  - IBM mainframe clone, but with some changes
- By the spring of 1985, this too was running a re-implemented EMAS, known as **EMAS-3**
  - This ran for several years

- By mid-1988, the EMAS-3 service had been moved to an (Itel, NAS, HITACHI) AS/VL-80
  - Large, powerful system with 8 front end PDP-11/73s
- In June 1992, the final EMAS-3 service was shut down

Timeline



- The project spanned 26 years
- User service spanned 22 years
- The Kent service spanned 7 years (on 2900 only)

- Apart from the EMAP era, the development team was always small

Part II – University of Kent

- From 1969 to 1976, Kent ran the Kent Online Service (KOS) on (initially) one Elliott 4130 with 64kW/96kW of memory
  - A second 4130 was added around 1974
  - This service mainly provided BASIC for teaching, with some editing and other utilities, plus a substantial batch service
- In 1976, these were replaced with an ICL 2960 running VME/K
- This was very unreliable; MTBF (combined hardware and software) around 20 hours
  - System fixes were received via badly Xeroxed hexadecimal patches
  - In the early days, 200 bug reports were submitted in a single month
- The system was not very user friendly
- 18 months later, ICL proposed to re-engineer VME/K, but would remove some features which were essential to the Kent workload

- It was time to investigate alternatives

- In April 1978, Peter Stephens, from Edinburgh, installed EMAS one evening, finishing in time for dinner!
- Bob Eager was the sole contact and systems person for EMAS itself
    - Others handled networking and applications

- Early testing showed a lock up in the EMAS Director (q.v.)
    - Traced to a bug which only showed up with numerous registered users (more than Edinburgh)
    - Fixed within hours
- The system was inexplicably a lot slower than expected
    - This was traced to a hardware incompatibility with the way things were done on other machines (2970, 2980) in the range
    - A software fix was applied at the operator console, on the running system!
- There were hardware (parity) errors on the link to the PDP-11/34 front end
    - This was traced to an earthing problem, due to the 2960 and 11/34 being on different power feeds
    - Fixed (probably with an earthing braid)

- The Kent network was based on a Cambridge ring
  - Differences were all handled inside the front-end processor
- User trials were run every day for two hours
- The changeover took place after the end of term in December 1979, and there was no going back
  - A utility was written to transfer users and files from the VME/K disks
- The MTBF went from 20 hours to around 2000 hours
  - EMAS was more resilient to hardware faults (particularly disk controller crashes)
- Initial distrust from ICL engineers, after which they were very much on board

- A serious problem in 1982 (microcode crash) turned out to be a hardware design error
  - Fixed by Bob Eager with a hand-crafted microcode patch!

- In summer 1983, a second OCP (CPU) was acquired from Government sources (1981 census) for the cost of transport
  - More memory was added (to 4MB)
- It worked straight away, but cross reporting between OCPs did not work
  - Another range incompatibility
  - Documentation from ICL was not forthcoming, and it was not known what bits had to be poked where!
- Solved by Bob reverse engineering the microcode to work out where and what those bits were
  - System modified to correct the cross reporting
- The service ran very successfully until shutdown in 1986

- Kent contributions:
  - Various system enhancements, such as printer accounting
  - Applications: BASIC, assembler, BCPL, some packages, another editor

Kent 2960 circa 1981

Part III - Implementation

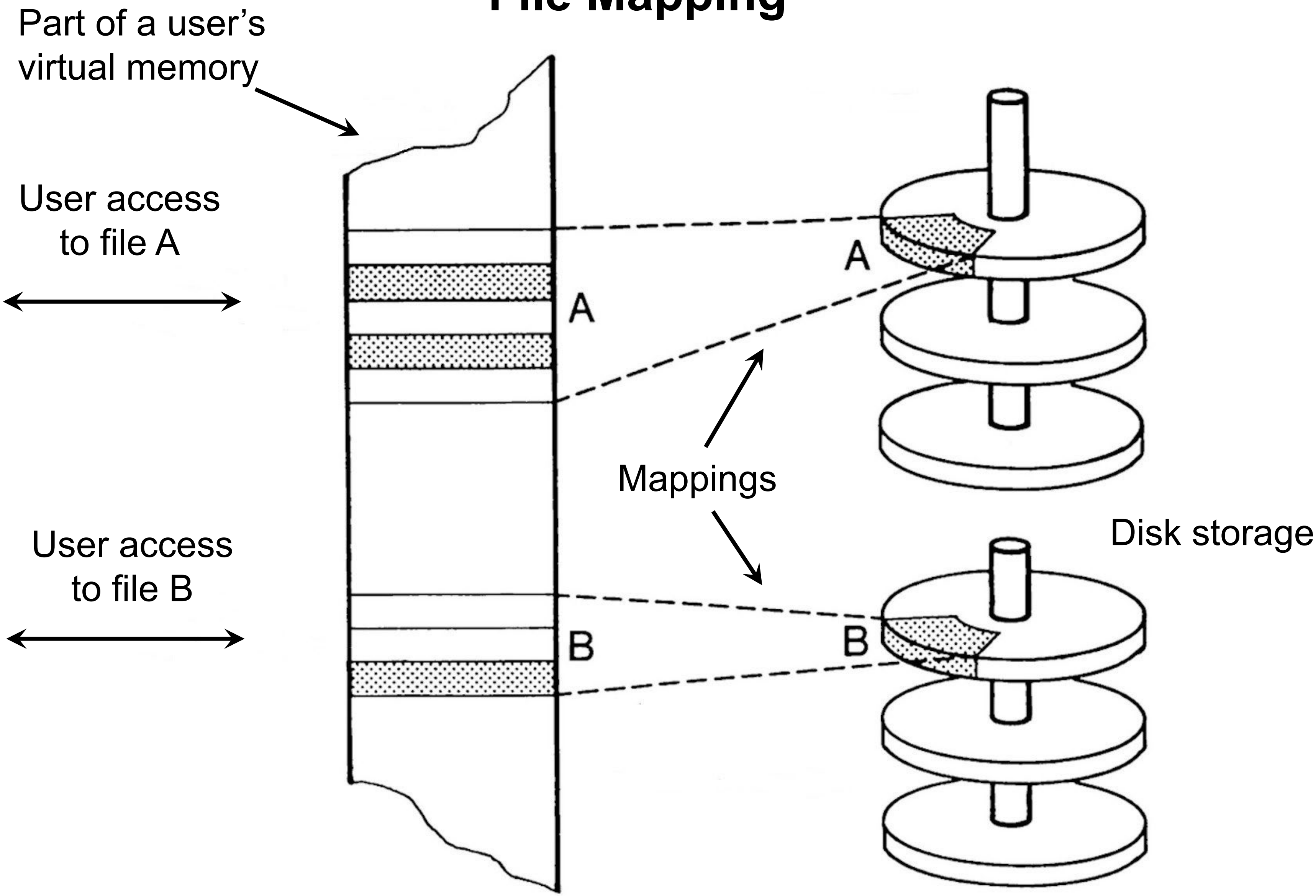
- Limited here mainly to the 2900 implementation, because:
  - 4/75 version (EMAS-1) was continually developed, and no sources are available
  - IBM/Amdahl/NAS (EMAS-3) based on EMAS 2900 anyway; incomplete sources available
  - 2900 sources are almost complete

- Information taken from papers and manuals, also from extensive reading of the source code!

Files and paging

- Files are accessed by *connecting* them into the virtual memory of a process; the user accesses them as if they are part of main storage
  - Connection is merely a book-keeping operation
  - There is no need for primitives such as *read*, *write*, or *seek*
- Files are divided into pages of fixed size; a page of a file is only actually brought into memory if it is referenced (either as code or data)
- Changed pages are automatically written back to disk when the memory is needed for something else, or on file *disconnection*
- Files can be shared by two or more processes, with only one copy of an active page being kept in memory
  - Code is thus automatically shared
  - File data (‘buffers’) can also be shared if a file has multiple users

File Mapping



The Kernel

- The system uses a small message passing kernel
- Messages are always 32 bytes, and are allocated from an expandable pool
  - 4-byte source, 4-byte destination, 24-byte payload
- Non-paged system processes include:
  - Device drivers
  - Paging manager
  - Semaphore implementation
  - Active memory handler
  - Scheduler
  - Interval timer and real time clock handler
  - Configuration control
  - Bulk storage mover (disk to disk, disk to tape, tape to disk)

- These are collectively known as the *global controller*
  - One instance per OCP (CPU)

Process structure

- Every non-resident process includes an instance of the *local controller* (q.v.), which is nonetheless *resident*, with shared code; it is privileged
  - Each process has its own local controller data and stack, although these are paged
- The privileged, but paged, portion of each non-resident process is supplied by the *Director*, again with shared code and per-process data and stack

- The non-privileged part of each process can vary, and is supplied by a *basefile*, connected in its virtual memory
  - This is all paged
- The standard basefile is the Edinburgh Subsystem, which provides the normal user interface
  - Users can supply their own basefile if desired
- Other basefiles exist, for example the Scientific Jobber (a fast batch compile/run system for student work)

1

Other processes

- Paged system processes provide various services:
  - DIRECT – operator control, logon, process start-up and other functions
  - VOLUMS – tape administration (also backup and archive)
  - SPOOLR – input and output spooling (local and remote)
  - FTRANS – file transfer (‘Blue Book’)
  - MAILER – electronic mail (‘Grey Book’)
- These all operate as user processes (for privileged users) with a specialised basefile (except DIRECT, which has no basefile)

The Local Controller

- The local controller is a resident part of each paged process, and (*inter alia*) handles paging *strategy* for that process (the global controller is responsible for the paging *mechanism*)
- Each process is allocated a page quota, which must not be exceeded; the quota changes over time according to process behaviour; enforced by the local controller
- Any ‘thrashing’ is thus local to one process; if it occurs, the process is rescheduled with more pages, and a different CPU profile
  - The algorithm is table driven, and easily changed
  - Batch processes effectively use a different table, with more emphasis on throughput and less on response time
- Each process is responsible for its own page replacement policy

- The quota system ignores page sharing, so memory can be under-used; solved by overcommitting (with tuned recovery in the event of deadlock)

The Director

- The Director is part of all paged processes, but has private data and stack for each process
- It implements:
  - File system code and data
  - System calls (passed to local controller via hardware call)
  - Contingency (exception) handling
  - Functions for the DIRECT process
- The Director is where the ‘security border’ is located; it runs at a less privileged level than the Local Controller, but is more privileged than ‘userland’
- The DIRECT process is unique in that it has *no basefile*; all of its code is inside the Director
- It is mainly responsible for operator interactions, as well as validating user logons, process start-up and other miscellaneous functions

File System Primitives (provided by Director)

- CONNECT – connect file in virtual memory
- DISCONNECT – disconnect file, write back pages

- CREATE – create a new file
- DESTROY – destroy a file
- RENAME – rename a file
- PERMIT – change permissions for a file
- GIVE FILENAMES – return names of files
- OFFER – offer a file to another user
- ACCEPT – accept an OFFERed file (disappears from donor)
- NEWGEN – overwrite a file with a newer version, even if in use; existing users retain the old file until disconnected

The Subsystem

- The subsystem is the ‘user’ part of a process; the *basefile* used by typical users of the system
- The usual subsystem is the *Edinburgh Subsystem*, but others can be built
- Runs in user (untrusted) mode
  - Mostly accessed by an interactive terminal
  - Can be used in batch, and there is a batch command language for job control
- A single process model is used (unlike, say, UNIX)
- Users can install additional commands from public files, or write and install their own for individual use

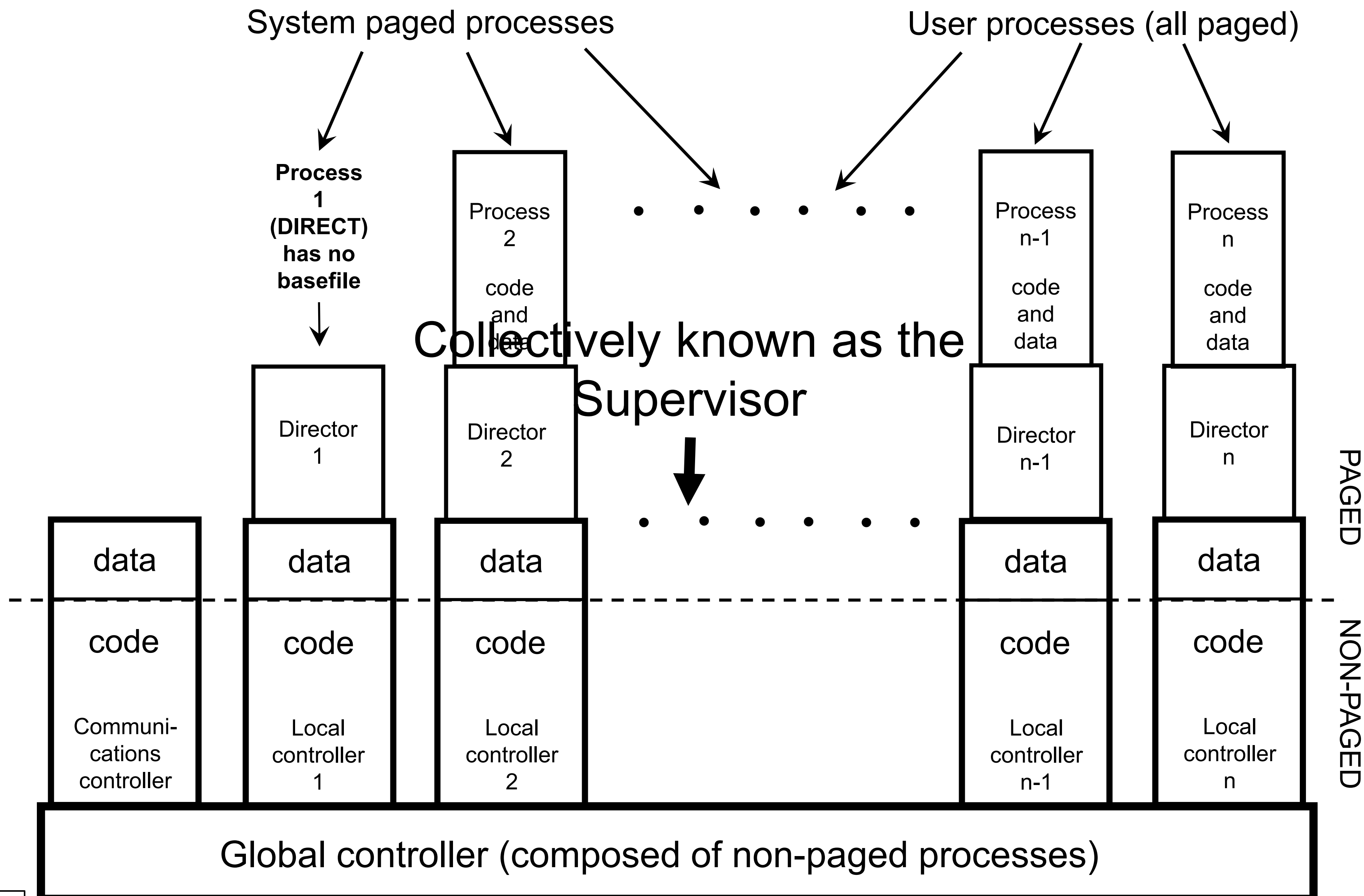
- Typical commands:
  - ANALYSE – give type specific information about a file
  - ALIAS – define alias for a command
  - ARCHIVE – queue file for archiving and removal from disk
  - OBEY – run a command script
  - FILES – list files in current directory
  - EDIT – edit a file (one of several editors)
  - COPY – copy a file
  - OPTION – set process options
  - CHERISH – mark file as ‘precious’
    - Will be backed up
    - Will be archived after four weeks of non-use (otherwise just deleted!)

  - OFFER, ACCEPT, PERMIT, RENAME, DESTROY, etc. – see calls to Director

- Files are just arrays of bytes, but the Subsystem suggests a minimal amount of information, and in practice this convention is almost always obeyed
- Files have a 32 byte header containing file type, date and time, physical size, logical size, record structure (if any) and pointers to other information such as object file data
- Basic ‘partitioned’ files store multiple files in one place; files can be copied freely into, and out of, partitioned files

- Subdirectories can also be created and used

System Structure



8

The Communications Controller

- The Communications Controller is a special version of the Local Controller, with its own page allocation
- It has no associated Director or basefile
- All I/O is from and to (parts of) files, which are connected (and paged in) as required
- I/O for terminals is via two terminal buffers, which are in a single one-page file
- This is mapped into the memory of the user process, and into the communications controller, as necessary

Initial Program Load (IPL, or boot)

- The boot program is large, and contains device drivers and code to interrogate what hardware is available (GROPE)
- It is essentially a chopped down version of the global controller, and is called …
  - CHOPSUPE!
- The hardware loads CHOPSUPE into memory starting at address zero
- Initial register settings are held within it, and they are loaded into the working registers before execution commences
- OCPs are queried for type, etc.; SMACs are queried for memory setup
- SACs are queried for trunks (controllers); controllers are queried for streams (devices)
- Commands are provided for disk formatting, system transfer from tape, etc., and then loading the actual system
- See later for the explanation of the initialisms

And that is (nearly) that …

Just one more thing…

Part IV – Resurrection?

- What would be needed to run EMAS again?

- Source code (no binaries available)
  - Source of EMAS (original) is probably now beyond reach
  - Source of EMAS 2900 has been preserved by Bob Eager
    - Includes base system, system processes, utility programs and some compilers
    - Missing are source of an assembler, and source of tape boot block
  - Source of EMAS-3 is patchy
  - Conclusion
    - EMAS 2900 would be the only option

- **Source code is the easy part; we have it**

1

- Assembler
  - Necessary to build EMAS bootstrap and tape boot block
  - No sources available
  - EMAS 2900 assembler was latterly the ICL assembler (MAPLE) with shims to make it work on EMAS; it was a binary file, and was not preserved
  - Possible, but impractical, to hand assemble

- **An assembler has been written and tested**
  - **~4000 lines of C**
  - **It assembles the 650 lines (350 statements) bootstrap**

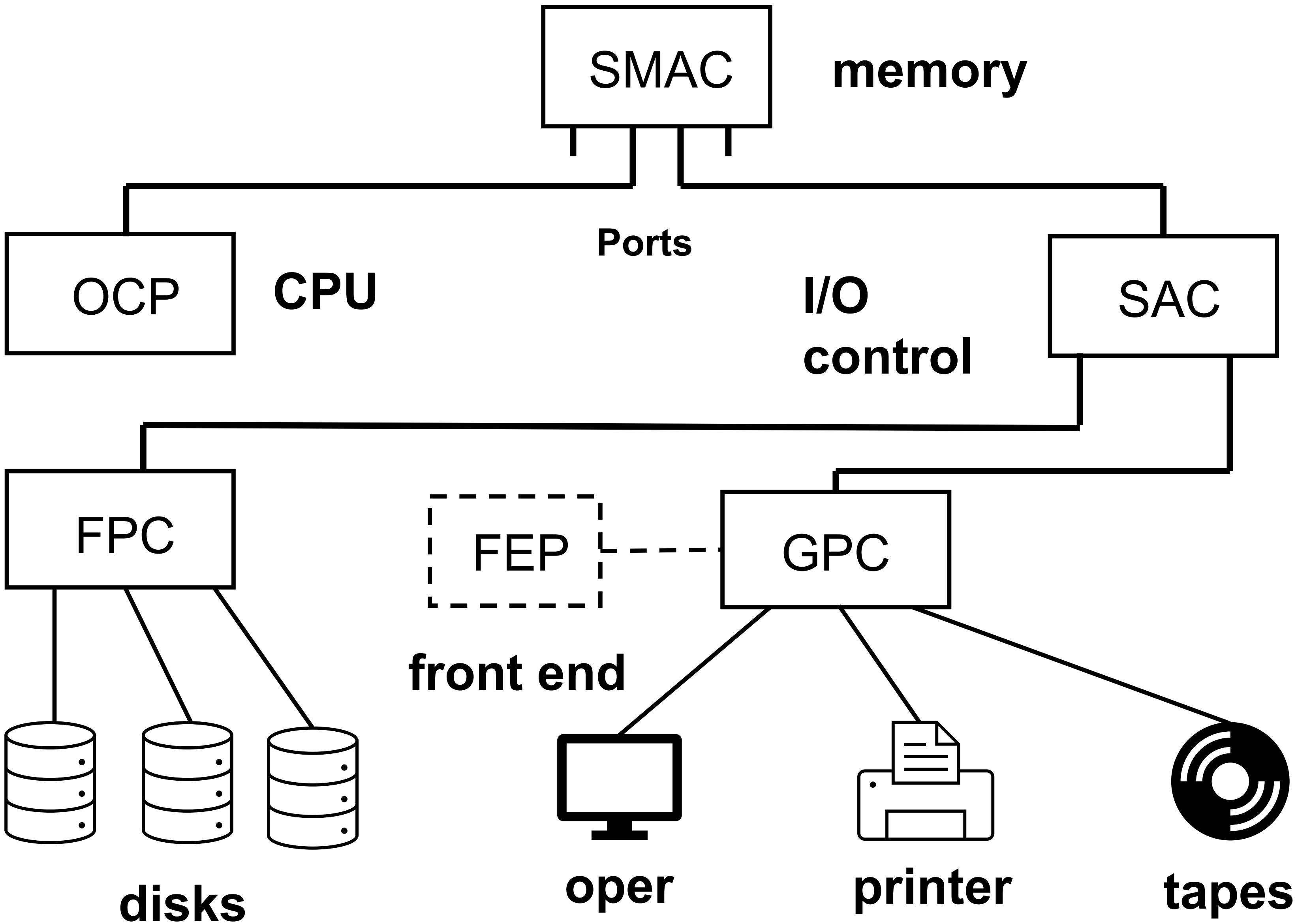
- IMP compiler
  - Would have to compile the IMP80 dialect, in which EMAS 2900 is written, into 2900 code
  - Source code (in IMP80) has been preserved; cannot be cross compiled by existing available compilers
    - The 2900 is *big-endian*; modern systems are generally *little-endian*
    - Uses IBM HFP floating point format
  - Would also require ancillary modules for object code output, and to provide a compiler environment (normally done by EMAS itself)

- **A compiler has been written and tested**
  - **~27,000 lines of C**

- Linker
  - Needed to link EMAS object modules to build system
  - Linker source code is available, but in IMP80
  - Requires library routines and the compiler environment provided by EMAS; has heavy dependence on the way EMAS handles files (as virtual memory)
- **Linker: a linker has been written and tested**
  - **~1,000 lines of C**
- Utility programs
  - To consolidate and fix up object files to make system and process images
  - EMAS has at least six of these, but four are quite similar
  - One is for CHOPSUPE, and is complicated, because it needs to generate segment tables and other data structures
  - Supervisor one (kernel/local controller) also has complications
- **Some utilities have already been written, as well as libraries for object code creation etc.**

- Hardware or simulator platform?
    - A simulator is the only practical path
- Two types of 2900 were supported by EMAS – the P series and the S series
    - Only P series considered here
- P series components required (minimum)
    - **S**tore **M**ultiple **A**ccess **C**ontroller – memory and interconnect
    - **O**rder **C**ode **P**rocessor – CPU
    - **S**tore **A**ccess **C**ontroller – autonomous I/O
    - **F**ile **P**eripheral **C**ontroller – disk controller
        - Disks (probably EDS100 or EDS200)
    - **G**eneral **P**eripheral **C**ontroller – interface to many peripherals
        - Tape drive
        - Operator station (OPER)
        - Printer

P series structure



- **A 2900 simulator is in progress, ~35,000 lines of C at present**
- So far:
  - OCP with all 128 instructions; interrupt control; segmentation and paging; clock, etc. supports 2960, 2970, 2972, 2976, 2980 (all P-series, single OCP only)
  - ECP (engineer's panel); minimal but sufficient
  - SMAC with configuration options, etc. (up to 8 units to get maximum memory)
  - SAC supporting autoconfiguration (single SAC only)
  - GPC supporting autoconfiguration (just one, easily expanded)
  - OPER with keyboard and two screens (only minimally tested so far)
  - Printer, for early diagnostic output (minimally tested)

- Still to be completed: tape drives; FPC; disk drives; card reader/punch; front end processor interface

- What works so far??
  - **CHOPSUPE loads and runs – up to a point**
    - Minor compiler error is the current issue
- How was it done?
  - Plenty of time
    - Beginnings of compiler in 1989, but mainly in 2021-22
    - Simulator started in 2002, but abandoned until 2021
- Why the delays?
  - Lack of documentation; all that was available were two partial documents on the OCP instructions
  - Unable to source documents on anything else
  - Most of the simulator has been developed by reverse engineering the EMAS source code

- Work continues …

And that really is the end …

Conclusion

- EMAS was a relatively long-lived system (22 years)
- It was shown to be very reliable
- It was extremely efficient at its job
  - The university environment is challenging but not necessarily mainstream
- It proved to be relatively portable
- Sadly, it was not widely used

- It is hoped that it will run again one day

Contact and further details

- Contact Bob at:
  - bob@eager.cx

- Bob's EMAS website
  - http://emas.bobeager.uk
  - Includes references and acknowledgements

- Documentation, source code, and many useful references:
  - http://www.ancientgeek.org.uk/EMAS