Nucleus

Goal of this system?

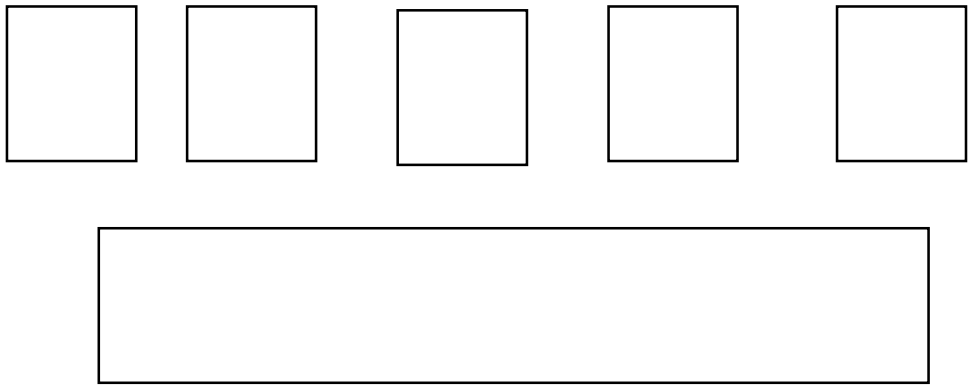
- Extensible

Goal of this system?

For the designer of advanced information systems, a vital requirement of any operating system is that it allow him to change the mode of operation it controls; otherwise his freedom of design can be seriously limited. Unfortunately, this is precisely what present operating systems do not allow.

This unfortunate situation indicates that the main problem in the design of a multiprogramming system is not to define functions that satisfy specific operating needs, but rather to supply a system nucleus that can be extended with new operating systems in an orderly manner. This is the primary objective of the RC 4000 system.

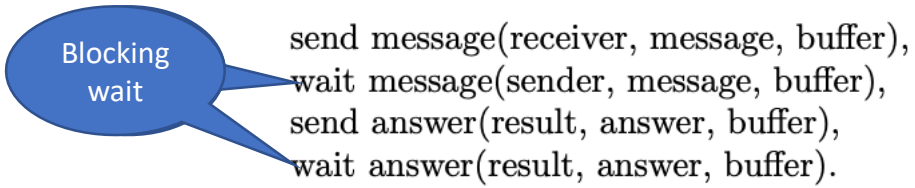
OS architecture: micro kernel



What is the minimum functionality a kernel should offer?

- Process creation (very different from THE!)
  - Start, Stop, Remove

- Process communication (very different from THE!)
  - A lot of security protection here!



Implementing different policies using nucleus mechanisms

- How to implement different scheduling policy?

Implementing different policies using nucleus mechanisms

- How to implement different memory management policy?

THE vs. Nucleus

	THE	Nucleus
Multi-programming?	Yes	Yes
dynamic processes?	No	yes
Memory management	VM	X
Synchronization	Semaphore	Message passing
OS architecture	Monolithic	Micro-kernel
Reliability		Better
Extensibility		Better
Performance	Better	

Multics

Background

- Multics
  - Multiplexed Information and Computing Service
- Unix
  - Uniplexed Information and Computing Service
- Many great inventions
  - Virtual memory
  - Security protection
  - File system

Goals

- “so the system would effectively serve the computing needs of a large community of users with diverse interests, operating principally from remote terminals”

(1) To provide the user with a large machine-independent virtual memory, thus placing the responsibility for the management of physical storage with the system software. By this means the user is provided with an address space large enough to eliminate the need for complicated buffering and overlay techniques. Users, therefore, are relieved of the burden of preplanning the transfer of information between storage levels, and user programs become independent of the nature of the various storage devices in the system.

Goal

(2) To permit a degree of programming generality not previously practical. This includes the ability of one procedure to use another procedure knowing only its name, and without knowledge of its requirements for storage, or the additional procedures upon which it may in turn call.

(3) To permit sharing of procedures and data among users subject only to proper authorization. Sharing of

(1) To provide the user with a large machine-independent virtual memory, thus placing the responsibility for the management of physical storage with the system software. By this means the user is provided with an address space large enough to eliminate the need for complicated buffering and overlay techniques. Users, therefore, are relieved of the burden of preplanning the transfer of information between storage levels, and user programs become independent of the nature of the various storage devices in the system.

Outline

- Virtual memory background
- Virtual memory in Multics
- Data sharing
- Code sharing

Backround

Virtual address, physical address, address space

Benefits of virtual memory

- Programming becomes easy
- More portable across machines
- Easier physical memory management
  - No need of contiguous allocation
  - Large virtual address space on small physical memory
- Benefit for MULTICS particularly
  - Code sharing, data sharing (dynamic link)

Segmentation (today)

- How many segments?
  - 3 segments
  - Code, heap, stack

- How to translate VA to PA?

Paging (today)

- Page frame
- Page
- Page table

Segment vs. Paging

- What is the benefit of paging over segment?

- What is the benefit of segment over paging?