THE

background
-workload: a continuous flow of user programs as a service to U
- machine
  memory/core: 27-bit, 32K, 2.5micro-sec
  drum: 512K, 1024 w per track, rev. time 40m sec
  indirect addressing (good for stack)
  intterupt
  i/os: 3, 3 paper tape puncher, 2 teleprinter, 1 plotter, 1 line printer

benefit of multiprogramming (vs. no multi…)
+ reduce turn-around time
+ better use of peripheral devices
+ backing store & cpu
+ accomodate low-demanding tasks

Storage/memory mgmt
segment (i.e., today's page; content of a page); segment id is like virtual page number
page, core pages, drum pages (i.e., today's page frame): physical address
segment variable: page table
advantage:
+ no need to write back to the same drum page
+ a program has no need to occupy consecutive drum pages
+ make programming easier

process & synchronization
 process in THE is more like a concurrent execution abstraction
 each process is an independent stream of execution
 synchronized through synchronization operation (semaphore!)
 not assuming any speed (this is the key to parallel programming!)

organization
 level 0: processor management/virtualization
          timer interrupt allows: priority, quick response, no monopolize
          above this, no sense that cpu is shared (or there is only one cpu!)
  level 1: segment controller (memory management), drum interrupt
          above this, only reference virtual segment, not drum pages
  level 2: message interpreter, manage console keyboard
          above this, each process had its private virtual console
          at this level, users specifies which process it is addressing toh
          (level 2 uses level 1 for dictionary)
  level 3: input, output device management

  level 4: user programs

how this helps testing! :)

interesting:
* timely research and development efforts
* a machine w/ sound basic characteristics: interrupt system!
* system correctness proof
* 6 half-time people
* only for Algol programs (not for machine-language programs)
* non-deterministic bugs caused by interrupts
  "This decision, inspired by fear, is .. main contribution to the art of system design"

* semaphore
  P, V
  mutual exclusion
  private semaphore (ordering)

Nucleus

multi-programming (dynamic)
+
customized/diverse policies
 different scheduling (priority, fair-share, real-time)

=>provide a nucleus that can be extended

process
internal process (resource management unit, execution unit)
  interruptable program executed in a given storage area, with a name
  vs. program (static)
external process
  I/O, device drivers, timer
nucleus
  handle hardware (interrupt system, storage protection, communication, process support)

message-passing
sendmsg(msg, rcv, &buffer);// return immediately after getting buffer (perf!)
waitmsg(&msg, &sender, &buffer); //blocking, FIFO
sendans(result, msg, buffer);
waitans(&result, &msg, buffer);
against malicious sender
buffer-binding (w/ sender, rcv id, authentication!)
against missing rcvers
against resource exhaustion

external process
  just like internal, with more capability
  can be created
  message passing between ex- and in- accomplish tasks
    disk read/write
    timer

internal process

process creation
 modern:
  start, ready, run, blocked, zombie, die
 nucleus API:
  start, stop, remove
 resource mgmt:
  storage (part of parent)
  buffer (part of parent)
  memory & disk (part of parent)

How is protection achieved?
  Nucleus checks the parameter of process creation
  + h/w support (tagged memory)

How is resource management policy supported?
  1. memory size decided by parent process
  2. cpu time (mechanism: underlying round-robin; policy can be adjusted by parent processes through stop/start API)
  3. how to support overlapped memory (virtual memory)
     stop (A); out (A); in (B); start(B)

How to provide mechanism that allows different policies

==========

	THE	Nucleus
background		
harware support	interrupt, devices	tagged memory, interrupt
multi-programming	yes, static	yes, dynamic yes vs. no process static vs. dynamic
memory mgmt	virtual	?
organization	monolithic	micro-kernel
	layered-virtualization	policy-mechanism
	less reliable	more reliable
	less extensible	more extensible
	better perf.	worse perf.
synchronization	semaphore	msg-passing