Notes for November 17, 1999

1. Greetings and Felicitations!
2. Puzzle of the Day
3. Privilege in OSes
   a. None (original IBM OS; protect with password, or anyone can read it)
   b. Fence, base and bounds registers; relocation
   c. Tagged architectures
   d. Memory management based schemes: segmentation, paging, and paged segmentation
4. Different forms of access control
   a. UNIX method
   b. ACLs: describe, revocation issue
   c. MULTICS rings
5. MULTICS ring mechanism
   a. MULTICS rings: used for both data and procedures; rights are REWA
   b. (b_1, b_2) access bracket - can access freely; (b_3, b_4) call bracket - can call segment through gate; so if *a*'s access bracket is (32,35) and its call bracket is (36,39), then *assuming permission mode (REWA) allows access*, a procedure in:
      rings 0-31: can access *a*, but ring-crossing fault occurs
      rings 32-35: can access *a*, no ring-crossing fault
      rings 36-39: can access *a*, provided a valid gate is used as an entry point
      rings 40-63: cannot access *a*
   c. If the procedure is accessing a data segment *d*, no call bracket allowed; given the above, *assuming permission mode (REWA) allows access*, a procedure in:
      rings 0-32: can access *d*
      rings 33-35: can access *d*, but cannot write to it (W or A)
      rings 36-63: cannot access *d*
6. Capabilities
   a. Capability-based addressing: show picture of accessing object
   b. Show process limiting access by not inheriting all parent's capabilities
   c. Revocation: use of a global descriptor table
7. Lock and Key
   a. Associate with each object a lock; associate with each process that has access to object a key (it's a cross between ACLs and C-Lists)
   b. Example: use crypto (Gifford). X object enciphered with key K. Associate an opener R with X. Then:
      OR-Access: K can be recovered with any D_i in a list of n deciphering transformations, so
      $R = (E_1(K), E_2(K), ..., E_n(K))$ and any process with access to any of the D_i's can access the file
      AND-Access: need all n deciphering functions to get K: $R = E_1(E_2(...E_n(K)...))$