Garbage Collection
Tutorial Outline

• Review of exercise 2
• Review of lectured material
• Discussion: real-time garbage collection
Review of Exercise 2

- Consider a system of periodic tasks: \( T_1 = (3, 1) \), \( T_2 = (4, 0.5) \), \( T_3 = (10, 2) \). The system must support three aperiodic jobs:
  - \( A_1 \) is released at time 0.5
  - \( A_2 \) is released at time 12.25
  - \( A_3 \) is released at time 17

- The aperiodic jobs execute for 0.75 units of time. The system is scheduled using the rate monotonic algorithm, with a simple sporadic server \( T_s = (5, 0.5) \) supporting the aperiodic jobs.

- Simulate the system for sufficient time to show the rate monotonic schedule and how the aperiodic jobs are scheduled. What is the response time for each of the aperiodic jobs?
Review of Exercise 2: Worked Answer

1) C1; R2 ⇒ t_e = MAX(t_r, BEGIN) = 0; replenish at t_e+p_s = 5
2) Replenished due to previous R2; executes according to C1
   R2 ⇒ t_e = t_r = 5 since END < t_r; replenish at t_e+p_s = 10
3) Job A_1 ends, but T_s continues according to C2
4) Replenished early due to R3(b)
5) C1; R2 ⇒ t_e = MAX(t_r, BEGIN) = 12; replenish at t_e+p_s = 17
6) Budget exhausted (R3(a) does not apply, already replenished at step 4)
7) Replenished early due to R3(b)
8) C1; R2 ⇒ t_e = MAX(t_r, BEGIN) = 15; replenish at t_e+p_s = 20
9) C2
10) Replenished early due to R3(b)
11) C1; R2 ⇒ t_e = MAX(t_r, BEGIN) = 18; replenish at t_e+p_s = 23
12) Replenished early due to R3(b)
13) C1

A1 : 0.5 → 5.25 response time = 4.75
A2 : 12.25 → 16.75 response time = 4.5
A3 : 17.0 → 20.75 response time = 3.75
Review of Lectured Material

• Automatic memory management
  • Stack allocation

• Reference counting
  • Simple, incremental, problems with cycles

• Garbage collection
  • Mark-sweep
  • Mark-compact
  • Copying collectors
  • Generational collectors
  • Real-time collectors

• Practical factors
Key Learning Outcomes

- Concepts of automatic memory management
- Reference counting: what, when, and why?
- Garbage collection concepts
  - Basic mark-sweep algorithm
  - Limitations, and rationale for copying collectors
  - Generational collectors: concepts, advantages and disadvantages
  - Incremental collectors
    - Tricolour marking
    - Read- and write-barriers
    - For real-time use
  - Practical limitations
Discussion: Real-time Garbage Collection

• Problems with prior work
  • Fragmentation and inability to handle large data structures
  • High-space overhead
  • Uneven mutator (program) utilisation: garbage collector consumes significant fraction of available CPU time

• Basic operation of the real-time collector
  • Free lists for different size blocks
  • Non-copying (mostly) - arraylets
  • Incremental mark-sweep algorithm, with read barrier
  • Occasional copies, for defragmentation

• Real-time scheduling
  • Analytical analysis to show performance bounds
  • Practical factors and implementation issues