CS5600 Computer Systems
Homework Assignment 02: Memory

Hang Zhao
NUID: 002826538

February 2, 2026

Problem 1: Memory Allocation [60 points]

Given Information

We have six memory partitions:

- Partition 1: 300 KB
- Partition 2: 600 KB
- Partition 3: 350 KB
- Partition 4: 200 KB
- Partition 5: 750 KB
- Partition 6: 125 KB

We need to allocate processes in order:

- Process A: 115 KB
- Process B: 500 KB
- Process C: 358 KB
- Process D: 200 KB
- Process E: 375 KB

First-Fit Algorithm

The first-fit algorithm allocates each process to the first partition that is large enough.

Process A (115 KB):

- Check Partition 1 (300 KB): $300 \geq 115$ ✓
- Allocate to Partition 1
- Remaining space in Partition 1: $300 - 115 = 185$ KB

Process B (500 KB):

- Check Partition 1 (185 KB): $185 < 500$ ×
- Check Partition 2 (600 KB): $600 \geq 500$ ✓
- Allocate to Partition 2
- Remaining space in Partition 2: $600 - 500 = 100$ KB

Process C (358 KB):

- Check Partition 1 (185 KB): $185 < 358$ ×
- Check Partition 2 (100 KB): $100 < 358$ ×
- Check Partition 3 (350 KB): $350 < 358$ ×
- Check Partition 4 (200 KB): $200 < 358$ ×
- Check Partition 5 (750 KB): $750 \geq 358$ ✓
- Allocate to Partition 5
- Remaining space in Partition 5: $750 - 358 = 392$ KB

Process D (200 KB):

- Check Partition 1 (185 KB): $185 < 200$ ×
- Check Partition 2 (100 KB): $100 < 200$ ×
- Check Partition 3 (350 KB): $350 \geq 200$ ✓
- Allocate to Partition 3
- Remaining space in Partition 3: $350 - 200 = 150$ KB

Process E (375 KB):

- Check Partition 1 (185 KB): $185 < 375$ ×
- Check Partition 2 (100 KB): $100 < 375$ ×

- Check Partition 3 (150 KB): $150 < 375$ ×
- Check Partition 4 (200 KB): $200 < 375$ ×
- Check Partition 5 (392 KB): $392 \geq 375$ ✓
- Allocate to Partition 5
- Remaining space in Partition 5: $392 - 375 = 17$ KB

First-Fit Summary:

Partition	**Original Size**	**Process Allocated**	**Remaining Space**
1	300 KB	A (115 KB)	185 KB
2	600 KB	B (500 KB)	100 KB
3	350 KB	D (200 KB)	150 KB
4	200 KB	-	200 KB
5	750 KB	C (358 KB), E (375 KB)	17 KB
6	125 KB	-	125 KB

Total wasted space: $185 + 100 + 150 + 200 + 17 + 125 = 777$ KB

Best-Fit Algorithm

The best-fit algorithm allocates each process to the smallest partition that can fit it.

Process A (115 KB):

- Available partitions: 300, 600, 350, 200, 750, 125 KB
- Partitions that fit: 300, 600, 350, 200, 750, 125 KB
- Smallest that fits: 125 KB
- Allocate to Partition 6
- Remaining: $125 - 115 = 10$ KB

Process B (500 KB):

- Available partitions: 300, 600, 350, 200, 750, 10 KB
- Partitions that fit: 600, 750 KB
- Smallest that fits: 600 KB
- Allocate to Partition 2
- Remaining: $600 - 500 = 100$ KB

Process C (358 KB):

- Available partitions: 300, 100, 350, 200, 750, 10 KB
- Partitions that fit: 750 KB (350 KB is too small: $350 < 358$)
- Allocate to Partition 5
- Remaining: $750 - 358 = 392$ KB

Process D (200 KB):

- Available partitions: 300, 100, 350, 200, 392, 10 KB
- Partitions that fit: 300, 350, 200, 392 KB
- Smallest that fits: 200 KB
- Allocate to Partition 4
- Remaining: $200 - 200 = 0$ KB

Process E (375 KB):

- Available partitions: 300, 100, 350, 0, 392, 10 KB
- Partitions that fit: 392 KB

- Allocate to Partition 5
- Remaining: $392 - 375 = 17$ KB

Best-Fit Summary:

Partition	Original Size	Process Allocated	Remaining Space
1	300 KB	-	300 KB
2	600 KB	B (500 KB)	100 KB
3	350 KB	-	350 KB
4	200 KB	D (200 KB)	0 KB
5	750 KB	C (358 KB), E (375 KB)	17 KB
6	125 KB	A (115 KB)	10 KB

Total wasted space: $300 + 100 + 350 + 0 + 17 + 10 = 777$ KB

Worst-Fit Algorithm

The worst-fit algorithm allocates each process to the largest available partition.

Process A (115 KB):

- Available partitions: 300, 600, 350, 200, 750, 125 KB
- Largest partition: 750 KB
- Allocate to Partition 5
- Remaining: $750 - 115 = 635$ KB

Process B (500 KB):

- Available partitions: 300, 600, 350, 200, 635, 125 KB
- Largest partition: 635 KB
- Allocate to Partition 5
- Remaining: $635 - 500 = 135$ KB

Process C (358 KB):

- Available partitions: 300, 600, 350, 200, 135, 125 KB
- Largest partition: 600 KB
- Allocate to Partition 2
- Remaining: $600 - 358 = 242$ KB

Process D (200 KB):

- Available partitions: 300, 242, 350, 200, 135, 125 KB
- Largest partition: 350 KB
- Allocate to Partition 3
- Remaining: $350 - 200 = 150$ KB

Process E (375 KB):

- Available partitions: 300, 242, 150, 200, 135, 125 KB
- None of these partitions can fit 375 KB
- **Process E cannot be allocated!**

Worst-Fit Summary:

Partition	Original Size	Process Allocated	Remaining Space
1	300 KB	-	300 KB
2	600 KB	C (358 KB)	242 KB
3	350 KB	D (200 KB)	150 KB
4	200 KB	-	200 KB
5	750 KB	A (115 KB), B (500 KB)	135 KB
6	125 KB	-	125 KB

Total wasted space: $300 + 242 + 150 + 200 + 135 + 125 = 1152$ KB
Note: Process E (375 KB) could not be allocated with worst-fit.

Comparison and Ranking

Summary of Results:

Algorithm	Processes Allocated	Total Wasted Space	Efficiency
First-Fit	All 5 processes	777 KB	Best
Best-Fit	All 5 processes	777 KB	Best
Worst-Fit	Only 4 processes	1152 KB	Worst

Ranking (Most to Least Efficient):

1. **First-Fit and Best-Fit (tie)**: Both successfully allocated all 5 processes with 777 KB of wasted space. They are equally efficient for this particular scenario.

2. **Worst-Fit**: Failed to allocate Process E and resulted in more fragmentation (1152 KB wasted). This is the least efficient algorithm for this workload.

Analysis:

The first-fit and best-fit algorithms performed equally well in this case, both managing to allocate all processes with the same amount of wasted space. This happened because the particular order and sizes of processes allowed both algorithms to make similar allocation decisions.

Worst-fit performed poorly because it tends to break up large partitions unnecessarily. By always choosing the largest partition, it prevented larger processes from being allocated later. Specifically, it used up Partition 5 (750 KB) early for small processes A and B, leaving no partition large enough for Process E (375 KB).

In general, best-fit tends to be more efficient than first-fit, but this depends heavily on the specific workload. Worst-fit typically performs the poorest because it causes severe fragmentation.

Problem 2: Scheduling [40 points]

Given Information

We have two processes with the following characteristics:

- Process P1: Period = 50, Execution time (t_1) = 25
- Process P2: Period = 75, Execution time (t_2) = 30

Under Earliest Deadline First (EDF) scheduling, the process with the earliest absolute deadline gets the highest priority. Priorities are assigned dynamically based on deadlines.

Deadline Calculation

For periodic processes, the deadline for each job is typically at the end of its period.

Process P1 deadlines:

- Job 1: Arrives at time 0, Deadline = 50
- Job 2: Arrives at time 50, Deadline = 100
- Job 3: Arrives at time 100, Deadline = 150

Process P2 deadlines:

- Job 1: Arrives at time 0, Deadline = 75
- Job 2: Arrives at time 75, Deadline = 150

Scheduling Analysis

Let me trace through the schedule step by step:

Time 0:

- P1 Job 1 arrives (deadline = 50)
- P2 Job 1 arrives (deadline = 75)
- Compare deadlines: $50 < 75$
- Schedule P1 first (earliest deadline)

Time 0-25:

- P1 Job 1 executes for 25 time units
- P1 Job 1 completes at time 25

Time 25:

- Only P2 Job 1 is ready (deadline = 75)
- Schedule P2

Time 25-50:

- P2 Job 1 executes for 25 time units (still needs 5 more)
- At time 50: P1 Job 2 arrives (deadline = 100)

Time 50:

- P2 Job 1 is running (deadline = 75, needs 5 more time units)
- P1 Job 2 arrives (deadline = 100)
- Compare deadlines: $75 < 100$
- Continue P2 (earliest deadline)

Time 50-55:

- P2 Job 1 completes remaining 5 time units
- P2 Job 1 completes at time 55

Time 55:

- Only P1 Job 2 is ready (deadline = 100)
- Schedule P1

Time 55-75:

- P1 Job 2 executes for 20 time units (still needs 5 more)
- At time 75: P2 Job 2 arrives (deadline = 150)

Time 75:

- P1 Job 2 is running (deadline = 100, needs 5 more time units)
- P2 Job 2 arrives (deadline = 150)
- Compare deadlines: $100 < 150$
- Continue P1 (earliest deadline)

Time 75-80:

- P1 Job 2 completes remaining 5 time units
- P1 Job 2 completes at time 80

Time 80:

- Only P2 Job 2 is ready (deadline = 150)
- Schedule P2

Time 80-100:

- P2 Job 2 executes for 20 time units (still needs 10 more)
- At time 100: P1 Job 3 arrives (deadline = 150)

Time 100:

- P2 Job 2 is running (deadline = 150, needs 10 more time units)
- P1 Job 3 arrives (deadline = 150)
- Deadlines are equal: both = 150
- Continue P2 (already executing, tie-breaking rule)

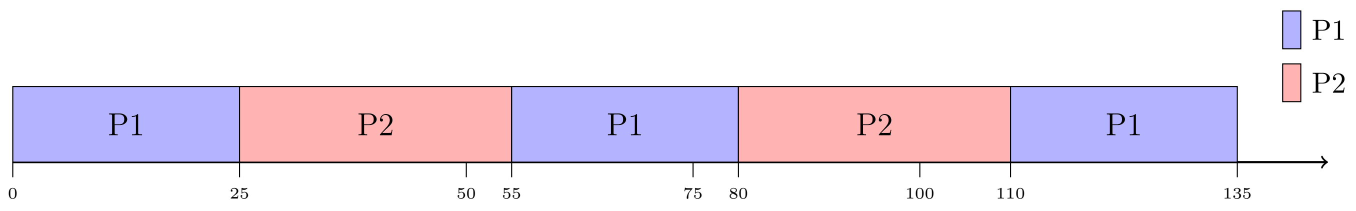
Time 100-110:

- P2 Job 2 completes remaining 10 time units
- P2 Job 2 completes at time 110

Time 110-135:

- P1 Job 3 executes for 25 time units
- P1 Job 3 completes at time 135

Gantt Chart



Verification

Let me verify that all deadlines are met:

Process P1:

- Job 1: Completes at time 25, Deadline = 50 (meets deadline)
- Job 2: Completes at time 80, Deadline = 100 (meets deadline)
- Job 3: Completes at time 135, Deadline = 150 (meets deadline)

Process P2:

- Job 1: Completes at time 55, Deadline = 75 (meets deadline)
- Job 2: Completes at time 110, Deadline = 150 (meets deadline)

Conclusion: All processes meet their deadlines under EDF scheduling. The schedule is feasible and optimal for this workload.