HW 3 Writeup

1. 9 processes (including the initial parent process).

2. Graph 1 (Left graph):
   Resources - R1, R2 and Threads - T1, T2, T3, T4
   T1, T3 hold R1 and request R2. T2, T4 hold R2 and request R1.
   This forms a circular wait condition, where processes are waiting indefinitely for resources held by each other.
   Since all four necessary deadlock conditions (mutual exclusion, hold and wait, no preemption, and circular wait) are satisfied, a deadlock exists in graph 1.

   Graph 2 (Right graph):
   Resources - R1, R2, R3 and Threads - T1, T2, T3, T4
   T1, T2 request R2, while T3 requests R1 and T4 requests R3.
   There is no circular dependency between the processes. No deadlock exists in graph 2 since there is no circular wait.

   Deadlock exists in Graph 1.

3. A - child: pid = 0
   B - child: pid1 = 2603
   C - parent: pid = 2603
   D - parent: pid1 = 2600

4. a) Order of execution:
   (i) P2 runs first and completes execution, t = 0 to 3
   (ii) P1 runs and completes execution, t = 3 to 11
   (iii) For t = 11 to 20, processor is idle
   (iv) P3 arrives at t=20 and completes execution, t = 20 to 24
   (v) From t = 24 to 25, processor is idle
   (v) P4 arrives at t=25, and completes execution t = 25 to 29
   (vi) From t = 29 to 45, processor is idle.
   (vii) At t = 45, P5 arrives and completes execution t = 45 to 50
   (viii) Processor idle from t = 50 to 55
   (ix) At t = 55, P6 arrives and completes execution t = 55 to 60

   Gantt chart:
   | P2 | P1 | Idle | P3 | Idle | P4 | Idle | P5 | Idle | P6 |
   0 3 11 20 24 25 29 45 50 55 60

   b) TAT for each process:
   TAT(P1) = 11 - 0 = 11
   TAT(P2) = 3 - 0 = 3
   TAT(P3) = 24 - 20 = 4

TAT(P4) = 29 - 25 = 4
TAT(P5) = 50 - 45 = 5
TAT(P6) = 60 - 55 = 5

c) Waiting time for each process:
WT(P1) = 11 - 8 = 3
WT(P2) = 3 - 3 = 0
WT(P3) = 4 - 4 = 0
WT(P4) = 4 - 4 = 0
WT(P5) = 5 - 5 = 0
WT(P6) = 5 - 5 = 0

5. b. Shortest job first and d. Priority based scheduling algorithms can lead to starvation.

6. In a hard real-time system, bounded interrupt and dispatch latency times are crucial for meeting timing constraints and ensuring the system can respond to critical events within a specified time frame.

   Interrupt latency is the time between when an interrupt is generated (e.g., from hardware) and when the system starts processing that interrupt. Interrupt latency must be bounded to ensure that the system can respond to critical events as quickly as possible. If the interrupt latency is too long, the system may not be able to meet the timing requirements for these events, resulting in serious consequences.

   Dispatch latency is the time between when a task is ready to run and when the CPU starts executing it. If dispatch latency is too high, a real-time task may miss its deadline, leading to system failure. Unbounded dispatch latency can reduce system determinism, making real-time guarantees impossible. High-priority tasks must preempt lower-priority tasks immediately in hard real-time systems.