**Samonorom Chum**
**Tuesday, April 15, 2025**
**EECE 4811/5811 Operating System Spring 2025**
**Professor Tseng**

# HW5

## 1. Priority Inversion and Priority Inheritance (3 pt)

- Find some credible resources to learn about Priority Inversion (which we also talked about in the first part of this course), Priority Inheritance (PIP) and Priority Protection (PPP).
- In the CPU scheduling algorithms that we learned, there is no priority associated with each task/job; however, in real-time OS, tasks have different priorities. We will learn how such a difference leads to more challenges in this question.
- **Problem setup:**

  **i.** You have a real-time system with three tasks:
  T1 (High-priority thread): Priority = 1
  T2 (Low-priority thread): Priority = 3
  T3 (Medium-priority thread): Priority = 2

  **ii.** T1 and T2 need to access the shared resource that is protected by a lock, while T3 doesn't

- **Q1:** Use your own words to describe and explain a scenario of "**priority inversion,**" using these three threads and associated priority. You need to describe arrival time, and which thread preempts which and how priority plays a role.

---

## Answer to Q1:

A **priority inversion** is when a high-priority thread (T1) is blocked by a low-priority thread (T2), but a medium-priority thread (T3) is able to run and extend the delay.

- At **time = 0**, low-priority thread, T2, starts running and enters the **critical section** (section of switching priority) and acquires a lock so no other thread can proceed.
- At **time = 1**, high-priority thread, T1, arrives. T2 has a higher priority than T1, meaning T1 should preempt T2, but since T2 still has the lock, T1 cannot proceed.
- At **time = 2**, a medium-priority thread, T3, arrives. T3 has higher priority than T2 and does not require a lock, so it preempts T2. T3 pushes T2 out of the CPU even though T2 is holding a resource that T1 is waiting for.
- **T1 is now blocked and delayed while T3 is running**, which causes **priority inversion**.

- **We will next calculate how long each thread executes in the presence of Priority Inheritance (PIP) and Priority Protection (PPP).**

**i. Assumptions:**

T2 (Low priority): Takes 5 units of time to complete its critical section that accesses the shared resource.
T3 (Medium priority): Takes 3 units of time to complete its execution – no access to the shared resource.
T1 (High priority): Takes 4 units of time to complete its critical section that accesses the shared resource.
T2 arrives at time = 0 and holds the lock initially
T1 arrives at time = 1
T3 arrives at time = 2

**ii. Q2: For each algorithm (Priority Inheritance Protocol (PIP), Priority Protection Protocol (PPP), answer the following five sub-questions:**

1. How much time does T2 spend executing in each protocol (PIP and PPP)?
2. In which protocol does T2 complete its critical section first?
3. What is the total execution time for all threads in each protocol?
4. In which protocol does T1 experience a delay due to priority inversion?
5. Explain how the two protocols (PIP and PPP) affect the scheduling of T3 in this scenario.

○ **Note: full credits rewards with complete explanation and description of the ordering of the threads. A diagram and/or timeline is encouraged.**

○ **Key concepts**: research, real-time OS, scheduling, priority

---

# Answer to Q2:

**Priority Inheritance Protocol (PIP):** If a high-priority thread is blocked by a low-priority one that is **holding a lock**, the lower one **inherits the higher priority** until it **releases** the lock.

**Priority Protection Protocol (PPP):** A thread **cannot enter** a critical section unless it has **priority ≥ ceiling** of that resource.

**Main Difference:** PIP = dynamic boost, PPP = static ceiling block

1. How much time does T2 spend executing in each protocol (PIP and PPP)?
    a. **T2 takes 5 units of time in both PIP and PPP**, due to T2 completing its critical section without being preempt.

2. In which protocol does T2 complete its critical section first?
    a. **T2 finishes at time = 5 for both PIP and PPP**.
    b. PIP gives T2, T1's priority, while PPP does not allow T1 into the critical section till T2 completes due to the priority ceiling limit.

3. What is the total execution time for all threads in each protocol?
    a. PIP and PPP both take **13 units** (time = 0 to 12).

4. In which protocol does T1 experience a delay due to priority inversion?
    **a. T1 experiences a delay in both PIP and PPP, but they are handled differently.**
    **b.** In PIP the priority inversion is mitigated because **T2 inherits T1's priority** so T3 cannot start.
    **c.** In PPP, priority inversion is prevented upfront, meaning T2 gets the resource only when it is safe and T3 cannot preempt **due to the lock**.

5. Explain how the two protocols (PIP and PPP) affect the scheduling of T3 in this scenario.
    a. In PIP T3 is blocked until T1 finishes due to **dynamic priority boost**.
    b. In PPP T3 is blocked until T1 finishes due to **access restrictions**.

| **Execution Timeline Table (PIP vs PPP)** | | | | |
|---|---|---|---|---|
| **Time** | **PIP: Running Thread** | **Reason (PIP)** | **PPP: Running Thread** | **Reason (PPP)** |
| 0 | T2 | T2 arrives and acquires lock | T2 | T2 arrives and acquires lock |
| 1 | T2 | T1 arrives but is blocked on lock | T2 | T1 arrives but is blocked on lock |
| 2 | T2 **(inherits T1's priority)** | T3 arrives but cannot preempt boosted T2 | T2 | T3 arrives but cannot preempt T2 in critical section |
| 3 | T2 | Continues executing under inherited priority | T2 | Continues critical section |
| 4 | T2 | Still in critical section | T2 | Still in critical section |
| 5 | T2 | **Finishes** and releases lock; priority drops | T2 | **Finishes** and releases lock |
| 6 | T1 | Lock is free; T1 enters critical section | T1 | Lock is free; T1 enters critical section |
| 7 | T1 | Continues | T1 | Continues |
| 8 | T1 | Continues | T1 | Continues |
| 9 | T1 | **Completes** | T1 | **Completes** |
| 10 | T3 | Now runs after T1 is done | T3 | Now runs after T1 is done |
| 11 | T3 | Continues | T3 | Continues |
| 12 | T3 | **Completes** | T3 | **Completes** |

## 2. Virtual and Physical Addresses (3 pt)

○ Read Ch. 18 https://pages.cs.wisc.edu/~remzi/OSTEP/vm-paging.pdf

○ For each of the following hardware specifications and software configuration, state how many bits are needed for:

**a) Virtual address:** number of bits in the VA for the OS.

**IE:** 32 bits in a 32-bit OS (Given).

**b) Physical address:** (Given) RAM size = **# GB** = **# * 1024$^3$** = **N**, **$\log_2$(N)** = PA bits.

**IE:** (Given) RAM size = 1 GB = 1 * 1024$^3$ = 1073741824, $\log_2$(1073741824) = 30 bits in PA.

**c) Virtual page number:** (VA – offset)

**d) Physical page number:** (PA – offset)

**e) Offset:** Given page size **# KB = # * 1096 = N,** $\log_2$(**N**) = offset bits.

**IE:** (Given) Page size = **4 KB**

**Option 1:** 4 * 1024 = 4096 bytes, $\log_2$(4096) = 12 bits for offset.

**Option 2:** 4 * 1024 = 4096, $\log_{10}$(4096)/$\log_{10}$(2) = 12 bits for offset.

**i.** 32-bit operating system, 4-KB pages, 1 GB of RAM
**ii.** 32-bit operating system, 16-KB pages, 2 GB of RAM
**iii.** 64-bit operating system, 16-KB pages, 16 GB of RAM

○ Note: you need to **show the calculation** to get full credits

○ Note2: you need to show **15 numbers**

○ **Key concepts**: page, virtual and physical address format

---

## Answer to #2:

### i. 32-bit operating system, 4-KB pages, 1 GB of RAM

**a) Virtual address:** (For 32-bit OS).
- **VA = 32 bits**.

**b) Physical address:** $\log_2$(RAM * 1024$^3$ [if in GB]).
- RAM = **1 GB** = 1 * 1024$^3$ = 1073741824,
- $\log_2$(1073741824) = **PA = 30 bits**.

**c) Virtual page number:** (VA – offset).
- 32 bits – 12 bits = **VPN = 20 bits**.

**d) Physical page number:** (PA – offset).

- 30 bits – 12 bits = **PPN = 18 bits**.

e) **Offset:** $\log_2$(Page size * 1024 [if in KB]).
   - Page size = **4 KB** = 4* 1024 =4096 bytes,
   - $\log_2$(4096) = **Offset = 12 bits**.

### ii. 32-bit operating system, 16-KB pages, 2 GB of RAM

a) **Virtual address:**
   - **VA = 32 bits**.

b) **Physical address:**
   - RAM = **2 GB** = 2 * $1024^3$ = 2147483648,
   - $\log_2$(2147483648) = **PA = 31 bits.**

c) **Virtual page number**: (VA – offset).
   - 32 - 14 = **VPN = 18 bits.**

d) **Physical page number**: (PA – offset).
   - 31 - 14 = **PPN = 17 bits.**

e) **Offset**:
   - Page size = **16 KB** = 16 * 1024 = 16384,
   - $\text{Log}_2$(13684) = **Offset = 14 bits.**

### iii.64-bit operating system, 16-KB pages, 16 GB of RAM

a) **Virtual address:**
   - **VA = 64 bits**.

b) **Physical address:**
   - RAM = **16 GB** = 16 * $1024^3$ = 17179869184,
   - $\log_2$(17179869184,) = **PA = 34 bits.**

c) **Virtual page number**: (VA – offset).
   - 64 - 14 = **VPN = 50 bits.**

d) **Physical page number**: (PA – offset).
   - 34 - 14 = **PPN = 20 bits**.

e) **Offset**:
   - Page size = **16 KB** = 16 * 1024 = 16384,
   - $\text{Log}_2$(13684) = **Offset = 14 bits.**

# References:


**[1]** L. Sha, R. Rajkumar, and J. P. Lehoczky, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization," *IEEE Transactions on Computers*, vol. 39, no. 9, pp. 1175–1185, Sept. 1990, doi: 10.1109/12.57058.

**[2]** ChatGPT, “Assistance with HW5 priority scheduling and paging concepts,” OpenAI, personal communication, Apr. 13, 2025.

**[3]** R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, Version 1.10, Chapter 18, 2023. [Online]. Available: https://pages.cs.wisc.edu/~remzi/OSTEP/

**[4]** Shem Louisy Jr., "Help with report formatting," personal communication, Apr. 13, 2025.