Operating System Part -2

क्रिया (Process) की परिभाषा: संचालन तंत्र (Operating System) में प्रक्रिया एक ऐसा प्रोग्राम होता है जो चल रहा होता है। जब कोई प्रोग्राम शुरू होता है, तो यह एक प्रक्रिया बन जाती है, जो CPU और मेमोरी का उपयोग करती है।

Process Concept :-

संचालन तंत्र (Operating System) में प्रक्रिया अवस्थाएँ (Process States) यह दर्शाती हैं कि कोई प्रक्रिया (Process) किस चरण में है–क्या वह चल रही है, तैयार है, या समाप्त हो चुकी है।

मुख्य प्रक्रिया अवस्थाएँ (Process States)

नवीन (New) → जब कोई नया प्रोग्राम शुरू होता है, तो वह नवीन प्रक्रिया बनती है।

तैयार (Ready) → प्रक्रिया CPU द्वारा चलाए जाने के लिए तैयार होती है। लेकिन CPU इसे अभी नहीं चला रहा होता।

चल रही (Running) → इस अवस्था में प्रक्रिया CPU द्वारा सक्रिय रूप से चल रही होती है।

प्रतीक्षा (Waiting) → जब कोई प्रक्रिया किसी संसाधन (जैसे इनपुट या फ़ाइल) का इंतजार कर रही होती है, तो इसे प्रतीक्षा अवस्था में रखा जाता है

समाप्त (Terminated) → जब प्रक्रिया अपना कार्य पूरा कर लेती है, तो यह बंद हो जाती है और सिस्टम से हटा दी जाती है।

Process Control Block :-

Process Control Block (PCB) ऑपरेटिंग सिस्टम में किसी भी प्रोसेस (कार्य) की जानकारी को संगठित और ट्रैक करने के लिए उपयोग किया जाता है। इसे प्रोसेस का आईडी कार्ड भी कह सकते हैं, क्योंकि इसमें उस प्रोसेस से जुड़ी हर ज़रूरी जानकारी होती है।

PCB में क्या-क्या होता है?

PCB में कई महत्वपूर्ण डेटा स्टोर होते हैं:

Process ID (PID) – हर प्रोसेस का एक यूनिक नंबर होता है जिससे ऑपरेटिंग सिस्टम उसे पहचानता है।

Process State – प्रोसेस की वर्तमान स्थिति क्या है? (जैसे Running, Ready, Waiting आदि)

CPU Registers – प्रोसेस के उपयोग किए गए रजिस्टर और उनके स्टेट्स।

Program Counter – प्रोसेस को आगे कौन-सा इंस्ट्रक्शन एक्सीक्यूट करना है?

Memory Management Information – प्रोसेस के लिए आवंटित मेमोरी की डिटेल्स।

I/O Information – प्रोसेस द्वारा उपयोग किए जा रहे इनपुट/आउटपुट डिवाइसेज़ की जानकारी।

Priority Information – प्रोसेस की प्राथमिकता (Priority) जिससे CPU तय करता है कि कौन सा प्रोसेस पहले चलना चाहिए।

PCB का काम क्या होता है?

जब कोई नया प्रोसेस बनता है, तो ऑपरेटिंग सिस्टम उसके लिए PCB तैयार करता है। यह प्रोसेस के हर स्टेप को ट्रैक करता है ताकि CPU सही तरीके से प्रोसेस को मैनेज कर सके।

उदाहरण के लिए:

अगर प्रोसेस को रोकना हो, तो PCB उसकी स्थिति स्टोर करता है ताकि बाद में उसे वहीं से शुरू किया जा सके।

अगर एक से ज्यादा प्रोसेस मौजूद हों, तो PCB उनकी प्राथमिकता तय करता है और CPU सही प्रोसेस को रन करता है

Schedulers :-

ऑपरेटिंग सिस्टम में Schedulers का मुख्य कार्य प्रोसेस को CPU में सही समय पर और सही तरीके से चलाना होता है। ये प्रोसेस के आवंटन (allocation) और शेड्यूलिंग (scheduling) का प्रबंधन करते हैं ताकि सिस्टम तेजी से और कुशलता से काम कर सके।

Schedulers के प्रकार:

ऑपरेटिंग सिस्टम में तीन प्रकार के Schedulers होते हैं:

Long-Term Scheduler (Job Scheduler)

यह तय करता है कि कौन-कौन से प्रोसेस मेमोरी में लोड किए जाएं।

यह प्रोसेस के आने की दर (Degree of Multiprogramming) को नियंत्रित करता है।

अगर यह धीमा हो, तो कम प्रोसेस सिस्टम में लोड होंगे, जिससे CPU खाली रह सकता है।

Short-Term Scheduler (CPU Scheduler)

यह CPU को अगला प्रोसेस देने का काम करता है।

यह बहुत तेज़ काम करता है, क्योंकि हर कुछ मिलीसेकंड में नए प्रोसेस को CPU में भेजना पड़ता है।

यह Scheduling Algorithms (जैसे FCFS, SJF, Round Robin) का उपयोग करता है।

Medium-Term Scheduler

यह Suspend और Resume करने का कार्य करता है।

अगर मेमोरी में बहुत ज्यादा प्रोसेस आ जाएं, तो कुछ प्रोसेस को सस्पेंड कर देता है ताकि सिस्टम ओवरलोड न हो।

बाद में जब मेमोरी में जगह मिले, तो इन प्रोसेस को फिर से चालू किया जाता है।

Scheduling Algorithm क्या है?

Scheduling Algorithm वह नियम (Rules) होते हैं, जिनका उपयोग Scheduler करता है यह तय करने के लिए कि कौन-सा प्रक्रिया (Process) CPU को कब और कितनी देर के लिए मिलेगी। इनका मुख्य उद्देश्य CPU का अधिकतम उपयोग, सिस्टम का सुचारू संचालन और प्रक्रियाओं का उचित निष्पादन सुनिश्चित करना होता है।

Scheduling Algorithms के प्रकार

1. First Come First Serve (FCFS)

👉 यह सबसे साधारण और आसान Scheduling Algorithm है।

👉 इसमें जो प्रक्रिया पहले आती है, उसी को पहले निष्पादित किया जाता है।

👉 किसी भी प्रकार की प्राथमिकता (Priority) नहीं दी जाती।

👉 इसे ऐसे समझें:

अगर किसी बैंक में लोग लाइन में खड़े हैं, तो सबसे पहले आए व्यक्ति का काम पहले होगा।

🔹 नुकसान:

2. Shortest Job First (SJF)

👉 इस Algorithm में सबसे छोटे समय वाली प्रक्रिया (Shortest Job) को पहले निष्पादित किया जाता है।

👉 इससे सिस्टम तेज़ी से काम करता है।

🔹 नुकसान:

3. Round Robin (RR)

👉 यह Multitasking के लिए बहुत उपयोगी Algorithm है।

👉 इसमें प्रत्येक प्रक्रिया को निश्चित समय (Time Quantum) के लिए CPU मिलता है।

👉 जब किसी प्रक्रिया का समय समाप्त हो जाता है, तो अगले प्रक्रिया को CPU मिल जाता है।

🔹 समझने के लिए उदाहरण:

मान लीजिए किसी क्लास में 5 स्टूडेंट्स हैं और टीचर हर स्टूडेंट को 5 मिनट पढ़ाने का समय देती है। फिर अगर किसी को ज़रूरत हो, तो उसे बाद में फिर से मौका मिलता है।

🔹 फायदे:

✅ Multitasking और टाइम शेयरिंग सिस्टम में बहुत अच्छा काम करता है।

✅ Waiting Time कम हो जाता है।

4. Priority Scheduling

👉 इसमें प्रत्येक प्रक्रिया को Priority (महत्व) के आधार पर CPU दिया जाता है।

👉 सबसे अधिक प्राथमिकता वाली प्रक्रिया सबसे पहले निष्पादित होती है।

👉 यह Preemptive (किसी प्रक्रिया को रोककर दूसरी शुरू करना) और Non-Preemptive दोनों तरीकों से काम कर सकता है।

- समझने के लिए उदाहरण:

अगर किसी हॉस्पिटल में मरीजों को इलाज करवाना है, तो सबसे गंभीर बीमारी वाले मरीज को पहले इलाज मिलेगा।

- नुकसान:

5. Multilevel Queue Scheduling

👉 इसमें विभिन्न प्रकार की प्रक्रियाओं को अलग-अलग कतारों (Queues) में बाँट दिया जाता है।

👉 प्रत्येक कतार (Queue) के लिए अलग-अलग नियम बनाए जाते हैं।

👉 Example:

Conclusion (निष्कर्ष)

✅ FCFS – सरल है, लेकिन लंबी प्रक्रियाओं के कारण धीमा हो सकता है।

✅ SJF – सबसे छोटा काम पहले होता है, पर लंबे कार्य रुक सकते हैं।

✅ Round Robin – सभी को बराबर CPU समय देता है।

✅ Priority Scheduling – महत्वपूर्ण कार्य पहले होते हैं, पर Starvation की समस्या हो सकती है।

✅ Multilevel Queue – प्रक्रियाओं को समूह में बाँटकर निष्पादन किया जाता है।

Preemptive और Non-Preemptive Scheduling क्या है?

जब किसी प्रक्रिया (Process) को CPU मिल जाता है, तो यह कैसे और कब CPU छोड़ेगी, इसे दो तरीकों से नियंत्रित किया जाता है:

1. Preemptive Scheduling

👉 इसमें CPU किसी प्रक्रिया को बीच में रोककर दूसरी प्रक्रिया को दे सकता है।

👉 जब कोई महत्वपूर्ण (High Priority) प्रक्रिया आती है, तो मौजूदा प्रक्रिया को रोक दिया जाता है।

👉 Interrupt (रुकावट) आने पर CPU किसी अन्य प्रक्रिया को दे दिया जाता है।

👉 ऑपरेटिंग सिस्टम को यह निर्णय खुद लेना पड़ता है कि किसे CPU पहले दिया जाए।

उदाहरण:

🚦 किसी चौराहे पर ट्रैफिक लाइट लगी हुई है।

अगर एम्बुलेंस आती है, तो सामान्य वाहनों को रोक दिया जाता है और एम्बुलेंस को पहले जाने दिया जाता है।

इसी तरह, ऑपरेटिंग सिस्टम में महत्वपूर्ण कार्यों को पहले CPU मिलता है।

फायदे:

✅ High Priority Tasks को जल्दी पूरा किया जा सकता है।

✅ Multitasking के लिए अच्छा होता है।

✅ CPU का ज्यादा अच्छा उपयोग होता है।

नुकसान:

❌ अधिक Switching के कारण सिस्टम स्लो हो सकता है।

❌ प्रक्रिया बार-बार रुकने से किसी भी कार्य को पूरा करने में अधिक समय लग सकता है।

2. Non-Preemptive Scheduling

👉 इसमें एक बार CPU किसी प्रक्रिया को मिल जाता है, तो वह पूरा होने तक चलता रहता है।

👉 जब तक प्रक्रिया पूरी नहीं होती, तब तक उसे रोककर दूसरी प्रक्रिया शुरू नहीं की जाती।

👉 कोई बाहरी Interrupt नहीं होता।

👉 यह आसान और स्थिर तरीका है।

उदाहरण:

🎬 मूवी थिएटर में मान लीजिए एक फिल्म चल रही है।

अगर आपने फिल्म शुरू कर दी, तो वह पूरी समाप्त होने के बाद ही अगली फिल्म चलेगी।

इसी तरह, CPU एक प्रक्रिया पूरी होने तक उसे नहीं छोड़ता।

फायदे:

✅ कोई Switching Overhead नहीं होता, जिससे सिस्टम फास्ट चलता है।

✅ सिस्टम ज्यादा स्थिर (Stable) रहता है।

✅ प्रत्येक प्रक्रिया को पूरा होने तक पर्याप्त समय मिलता है।

नुकसान:

❌ Emergency Tasks को तुरंत CPU नहीं मिल सकता।

❌ कम Priority वाली प्रक्रियाएँ पहले आने पर High Priority कार्यों को इंतजार करना पड़ता है।

Important Terms :-

ऑपरेटिंग सिस्टम में प्रोसेस शेड्यूलिंग के दौरान Arrival Time, Burst Time, Completion Time, और Waiting Time जैसी महत्वपूर्ण शर्तें (Terms) होती हैं। ये टर्म्स किसी प्रोसेस के निष्पादन को समझने और समय का सही प्रबंधन करने में मदद करती हैं। चलिए विस्तार से समझते हैं:

1️⃣Arrival Time (आगमन समय)

👉 Arrival Time वह समय होता है जब कोई प्रोसेस Ready Queue में प्रवेश करता है और CPU द्वारा निष्पादन के लिए तैयार होता है।

👉 इसे टी (t) से दर्शाया जाता है और यह दर्शाता है कि प्रोसेस सिस्टम में कब आया।

- 🔹 उदाहरण:

अगर किसी प्रोसेस P1 का Arrival Time = 0ms है, तो इसका मतलब है कि यह प्रक्रिया स्टार्टिंग में ही सिस्टम में आ गई थी।

अगर P2 का Arrival Time = 3ms है, तो यह 3 मिलीसेकंड बाद सिस्टम में आया।

2️⃣ Burst Time (CPU Execution Time)

👉 Burst Time वह समय होता है जो किसी प्रोसेस को CPU पर चलने (Execution) में लगता है।

👉 इसमें प्रोसेस के द्वारा लिए गए CPU टाइम की गणना होती है।

👉 इसे पूरी तरह से प्रोसेस का रन टाइम कहा जा सकता है।

- 🔹 उदाहरण:

अगर प्रोसेस P1 को CPU पर 10ms तक चलने की जरूरत है, तो इसका Burst Time 10ms होगा।

अगर प्रोसेस P2 को 5ms लगते हैं, तो उसका Burst Time 5ms होगा।

3️⃣ Completion Time (समाप्ति समय)

👉 Completion Time वह समय होता है जब प्रोसेस पूरी तरह से निष्पादित (Executed) हो जाता है और सिस्टम से समाप्त हो जाता है।

👉 इसे यह समझने के लिए इस्तेमाल किया जाता है कि किसी प्रोसेस को कब पूरा किया गया।

- 🔹 उदाहरण:

अगर प्रोसेस P1 का Completion Time = 15ms है, तो इसका मतलब है कि यह 15 मिलीसेकंड बाद समाप्त हुआ।

Turnaround Time (TAT) क्या होता है?

Turnaround Time किसी प्रक्रिया (Process) की कुल समय अवधि होती है, यानी जब यह सिस्टम में आता है (Arrival Time) से लेकर जब यह पूरी तरह से समाप्त (Completion Time) होता है।

Turnaround Time का फॉर्मूला

[\text{Turnaround Time} = \text{Completion Time} - \text{Arrival Time}]

इसका अर्थ:

👉 जब कोई Process सिस्टम में आता है और अंत में पूरा होता है, उस समय को Turnaround Time कहते हैं।

Waiting Time (प्रतीक्षा समय) क्या होता है?

Waiting Time वह समय होता है जो किसी प्रक्रिया (Process) को CPU मिलने से पहले इंतजार करने में लगता है।

अगर कोई प्रोसेस पहले आ चुका है लेकिन उसे CPU नहीं मिला, तो जितनी देर वह इंतजार करेगा, वही उसका Waiting Time कहलाता है।

Waiting Time का फॉर्मूला

[\text{Waiting Time} = \text{Turnaround Time} - \text{Burst Time}]

इसका अर्थ:

👉 जब कोई प्रोसेस सिस्टम में आता है, उसे CPU प्राप्त करने से पहले जो समय इंतजार करना पड़ता है, वही उसका Waiting Time होता है।

1. Multilevel Queue Scheduling

Multilevel Queue Scheduling में, पूरी system के processes को उनके प्रकार के अनुसार अलग-अलग queues (कतारों) में बाँट दिया जाता है। उदाहरण के लिए:

- **Real-Time Processes:** ऐसे tasks जिनका execution तुरंत होना चाहिए, इनके लिए एक उच्च प्राथमिकता वाली queue।
- **Interactive Processes:** ऐसे tasks जो यूज़र के साथ interact करते हैं, इनके लिए भी एक अलग queue।
- **Batch Processes:** ऐसे tasks जो background में चलते हैं और execution में थोड़े समय की देरी भी चले, इनके लिए एक और queue।

मुख्य बातें:

- **Fixed Assignment:** हर process, एक बार उस specific queue में डाल दिए जाने के बाद, हमेशा उसी queue में रहता है। इसमें queue से process को दूसरे queue में स्थानांतरित नहीं किया जाता।
- **Queue के अंदर अलग Algorithms:** हर queue के अंदर एक-एक करके scheduling algorithm (जैसे कि FCFS, Round Robin आदि) चलाया जा सकता है, पर overall system में इन queues की प्राथमिकता पहले से तय रहती है।
- **Priority Based Execution:** System सबसे ऊँची प्राथमिकता वाली queue से processes को एक्सीक्यूट करता है। मतलब, अगर high-priority queue में कोई process है तो उसे पहले execute किया जाता है, भले ही दूसरी queue में process भी मौजूद हों।

2. Multilevel Feedback Queue Scheduling

Multilevel Feedback Queue Scheduling में भी processes को अलग-अलग queues में बाँटा जाता है, पर यहाँ एक खास फीचर है - **feedback mechanism**।

मुख्य विशेषताएँ:

- **Dynamic Assignment:** यहाँ processes को सिर्फ एक queue में लॉक नहीं किया जाता। यदि किसी process को ज्यादा CPU time लग रहा है (i.e., वह CPU-bound है) तो उसकी प्राथमिकता कम कर दी जाती है और उसे निम्न प्राथमिकता वाली queue में भेज दिया जाता है।
- **Priority Adjustment:** यदि कोई process कम CPU-समय लेता है (i.e., वह I/O-bound होता है या समय-समय पर wait करता है), तो उसे high-priority queue में वापस भेजा जा सकता है। इस तरह system dynamically निर्णय लेता है कि कौन से processes को पहले चलाना चाहिए।
- **Flexibility & Fairness:** ये व्यवस्था system को adaptable बनाती है, जिससे ज्यादा interactive या कम CPU-intensive processes को जल्दी response मिलता है और overall system performance सुधरती है।

3. Multiprocess Scheduling (या Multi-core Scheduling)

Multiprocess Scheduling का इस्तेमाल उन systems में किया जाता है जहाँ एक से अधिक processors (या multi-core CPUs) मौजूद होते हैं। इसमें मुख्य उद्देश्य है कि सभी processors का समान उपयोग हो और कोई processor idle न रहे।

मुख्य बिंदु:

- **Load Balancing:** Operating System processes को अलग-अलग processors में बाँटता है ताकि सभी processors पर समान रूप से लोड पड़े। इससे execution जल्दी और प्रभावी तरीके से होता है।
- **Scheduling Strategies:**
    - **Symmetric Multiprocessing (SMP):** सभी processors एक समान लचीला व्यवहार प्रदर्शित करते हैं। एक common queue से processes हर processor द्वारा उठाए जाते हैं।
    - **Asymmetric Multiprocessing:** प्रत्येक processor के लिए एक अलग queue हो सकती है और एक master processor यह तय करता है कि कौन सा process किस processor पर चलना चाहिए।
- **उद्देश्य:** multiprocess scheduling का उद्देश्य है कि system ke सभी cores optimal रूप से काम करें, जिससे overall performance बेहतर हो और किसी भी processor पर ज्यादा load ना पड़े।

यह समझना जरूरी है कि इन scheduling algorithms का design system की requirements और processes के प्रकार पर निर्भर करता है।

- **Multilevel Queue Scheduling** एक static, fixed approach है जहाँ processes कब और कहां execute होंगे, पहले से निर्धारित होता है।
- **Multilevel Feedback Queue Scheduling** में processes की priorities समय के साथ बदलती रहती हैं, जिससे dynamic behavior और fair distribution मिलता है।
- **Multiprocess Scheduling** specifically multi-core या multi-processor systems के लिए बनाई गई है, जिसमें work को efficiently distribute करने पर जोर होता है।

PROCESS SYSNCHRONIZATION

सिंक्रीकरण (Synchronization) का मतलब होता है कि कई प्रक्रियाएँ (processes) एक साथ, सही तालमेल के साथ काम करें। इसे ऐसे समझिए:

मान लीजिए, एक रसोई में कई लोग खाना बना रहे हैं। अगर वे बिना तालमेल के काम करें तो चीजें बिगड़ सकती हैं–कोई सब्जी काट रहा है लेकिन तड़का देने वाला तैयार नहीं, कोई गैस जला रहा है लेकिन पैन नहीं रखा।

लेकिन अगर सब तालमेल से काम करें–सब्जी काटने के बाद तुरंत तड़का लगे, फिर मसाले डालें, और आखिर में सर्व करें–तो खाना जल्दी और सही तरीके से तैयार होगा।

इसी तरह, कंप्यूटर में कई प्रक्रियाएँ चलती हैं। अगर सब अपना काम सही क्रम और तालमेल में करें, तो सिस्टम बेहतर तरीके से काम करता है। इस प्रक्रिया को ही "प्रोसेस सिंक्रीकरण" कहते हैं।

CRITICAL SECTION

मान लीजिए कि घर में एक ही बाथरूम है और कई लोग उसे इस्तेमाल करना चाहते हैं। अगर सभी लोग एक साथ अंदर जाने की कोशिश करें तो गड़बड़ हो जाएगी, है ना? इसलिए, जब एक व्यक्ति बाथरूम में होता है, बाकी लोग बाहर इंतजार करते हैं–ताकि कोई टकराव (conflict) न हो और सबका काम ठीक से हो सके।

इसी तरह, कंप्यूटर में भी कई प्रक्रियाएँ (processes) एक ही संसाधन (resource) को इस्तेमाल करना चाहती हैं, जैसे कोई डेटा या फाइल। अगर सभी एक साथ बदलाव करने लगें, तो डेटा खराब हो सकता है। इसलिए, "क्रिटिकल सेक्शन" वह हिस्सा होता है जहाँ कोई प्रक्रिया (process) अकेले काम करती है, ताकि बाकी से टकराव न हो।

समस्या को हल करने के लिए "सिंक्रीकरण" (Synchronization) के नियम बनाए जाते हैं, जिससे यह तय होता है कि एक समय में केवल एक ही प्रक्रिया "क्रिटिकल सेक्शन" में काम करे।

RACE CONDITION

मान लीजिए कि आप और आपका दोस्त एक ही दरवाजे से अंदर जाने की कोशिश कर रहे हैं। अगर दोनों एक साथ दौड़कर दरवाजे तक पहुँचते हैं और कोई तय नियम नहीं है कि कौन पहले जाएगा, तो हो सकता है कि आप टकरा जाएँ या दरवाजा अटक जाए।

इसी तरह, कंप्यूटर में जब दो या अधिक प्रक्रियाएँ (processes) एक ही संसाधन (resource) को एक साथ बदलने की कोशिश करती हैं और यह तय नहीं होता कि कौन पहले बदलाव करेगा, तो डेटा गलत हो सकता है या सिस्टम में गड़बड़ हो सकती है।

इस समस्या से बचने के लिए "सिंक्रीकरण" (Synchronization) तकनीकों का उपयोग किया जाता है, जिससे यह तय होता है कि कौन पहले काम करेगा और किसी भी टकराव (conflict) से बचा जा सके।

सेमाफोर (SEMAPHORE) को आसान भाषा में समझते हैं:

मान लीजिए, एक खेल के मैदान में झूला (Swing) लगा है, लेकिन उस पर एक बार में सिर्फ एक बच्चा झूल सकता है। अब, अगर कोई बच्चा झूला झूल रहा है, तो दूसरा बच्चा इंतजार करता है। जैसे ही पहला बच्चा झूला छोड़ता है, दूसरा बच्चा झूल सकता है।

इसी तरह, कंप्यूटर में सेमाफोर एक **सिंक्रोनाइज़ेशन मैकेनिज्म** है, जो यह तय करता है कि कितनी प्रक्रियाएँ (Processes) एक साथ किसी संसाधन (Resource) का उपयोग कर सकती हैं।

सेमाफोर के दो मुख्य ऑपरेशन होते हैं:

1️⃣ **Wait (या P ऑपरेशन)** – जब कोई प्रक्रिया संसाधन (Resource) का उपयोग करना चाहती है, तो वह पहले जाँचती है कि संसाधन खाली है या नहीं। अगर संसाधन उपलब्ध है, तो वह उसे पकड़ लेती है। अगर नहीं, तो उसे इंतजार करना पड़ता है।

2️⃣ **Signal (या V ऑपरेशन)** – जब कोई प्रक्रिया अपना काम पूरा कर लेती है, तो वह संसाधन को छोड़ देती है, ताकि दूसरी प्रक्रिया उसे इस्तेमाल कर सके।

- **उदाहरण:**

सोचिए, एक बैंक में सिर्फ 2 काउंटर खुले हैं। अगर कोई ग्राहक आता है, तो वह खाली काउंटर पर जा सकता है (Wait)। जब वह अपना काम पूरा कर लेता है, तो वह काउंटर खाली कर देता है (Signal), ताकि दूसरा ग्राहक उस पर आ सके।

इसी तरह, सेमाफोर कंप्यूटर सिस्टम में प्रक्रियाओं के बीच तालमेल बनाए रखने में मदद करता है।