Table of Contents

[A folyamat koncepció 1](#_Toc166790421)

[A folyamat állapotai 2](#_Toc166790422)

[Folyamat ütemezése 4](#_Toc166790423)

[d: Ütemezők 6](#_Toc166790424)

[Műveletek folyamatokon, folyamatok közötti kommunikáció 7](#_Toc166790425)

[Távoli eljáráshívások 8](#_Toc166790426)

[Üzenetátadás, csővezetékek 10](#_Toc166790427)

**3. Folyamatok**

# A folyamat koncepció

A folyamat koncepciója alapvető jelentőséggel bír az operációs rendszerek működésében. Egy folyamat egy aktív program végrehajtásának kontextusa, amely magában foglalja a programkódot, az aktuális tevékenységeket, a regiszterek állapotát, valamint a program által használt erőforrásokat, mint például a memória, nyitott fájlok, és I/O eszközök.

A Folyamat Alapvető Elemei

**Programszámláló** (Program Counter, PC): Tartalmazza a következő végrehajtandó utasítás memóriacímét.

**Regiszterek**: Az aktuális műveletekhez szükséges adatokat és állapotinformációkat tartalmazza.

**Memória menedzsment információ**: Tartalmazza azokat az információkat, amelyek a folyamat virtuális és fizikai memóriájának kezeléséhez szükségesek, beleértve a memória határait, oldaltáblákat, és a memória védelmi állapotát.

**Számlálók és időzítők**: A folyamat CPU használatát és rendszeridőt követik nyomon.

**I/O állapot információ**: Információ azokról az I/O eszközökről, amelyekhez a folyamat hozzáfér, beleértve a listát a nyitott fájlokról.

**Folyamat Állapotok**

A folyamat életciklusa során különböző állapotokon megy keresztül, amelyek között az operációs rendszer kezeli az átmeneteket:

Új (New): A folyamat létrejöttét és inicializálását jelzi.

Kész (Ready): A folyamat várakozik a CPU-ra, hogy futtatásra kerüljön.

Futó (Running): A folyamat aktívan használja a CPU-t.

Várakozó (Waiting): A folyamat egy eseményre vagy adatokra vár, mielőtt folytatódhatna.

Befejezett (Terminated vagy Exit): A folyamat befejezte a végrehajtását.

Folyamatok Kezelése

Az operációs rendszer a folyamatok kezeléséért felelős, beleértve a folyamatok létrehozását, ütemezését, szinkronizációját és kommunikációját. A folyamatok kezelése magában foglalja az erőforrások (például CPU idő, memória) hatékony elosztását és a folyamatok közötti konfliktusok feloldását.

Folyamatok és Szálak

A modern operációs rendszerek gyakran különbséget tesznek folyamatok és szálak között. A szálak, más néven könnyűsúlyú folyamatok, egy folyamaton belül futnak, és megosztják annak erőforrásait, de saját regiszterekkel és számlálókkal rendelkeznek. A szálak lehetővé teszik a folyamatok párhuzamos végrehajtását egyetlen alkalmazáson belül, javítva a teljesítményt többmagos processzorokon.

Összefoglalva, a folyamat koncepciója az operációs rendszer működésének szívében helyezkedik el, lehetővé téve a számítógép erőforrásainak hatékony és biztonságos kezelését, miközben támogatja az alkalmazások párhuzamos végrehajtását.

## A folyamat állapotai

A folyamat állapotai azokat a különböző fázisokat jelölik, amelyeken egy folyamat áthalad életciklusa során az operációs rendszerben. Ezek az állapotok segítenek az operációs rendszernek a folyamatok hatékony kezelésében, az erőforrások elosztásában, és a CPU ütemezésében. A leggyakoribb folyamatállapotok a következők:

1. Új (New)

Ebben az állapotban a folyamat létrejön. A rendszer inicializálja a folyamat szükséges erőforrásait és struktúráit, mint például memória és folyamatkontroll blokk (PCB).

2. Kész (Ready)

A kész állapotban lévő folyamatok készen állnak a végrehajtásra, és arra várnak, hogy a processzor időt kapjanak. Ebben az állapotban a folyamat már rendelkezik minden szükséges erőforrással a futáshoz, kivéve a CPU-t.

3. Futó (Running)

Amikor a folyamat CPU időt kap és ténylegesen végrehajtódik. Ebben az állapotban a folyamat utasításai futnak, és a folyamat aktívan dolgozik a feladatán.

4. Várakozó (Waiting vagy Blocked)

Egy futó folyamat akkor kerül várakozó állapotba, ha valamilyen esemény bekövetkezésére vagy adat elérhetőségére vár, például I/O művelet befejeződésére vagy egy erőforrás felszabadulására. A folyamat ebben az állapotban nem használja a CPU-t, és várakozik az esemény bekövetkezéséig.

5. Kész-várakozó (Ready to Run)

Ez egy olyan átmeneti állapot, amelyet néhány rendszer használ. Itt a folyamat már megkapta az I/O művelethez szükséges adatokat, és ismét kész állapotba kerül, hogy folytassa a végrehajtást.

6. Befejezett (Terminated vagy Exit)

Miután a folyamat befejezte végrehajtását, vagy ha azt explicit módon megszakították, a terminated állapotba kerül. Ebben az állapotban a folyamat erőforrásait felszabadítják, és a folyamatkontroll blokk (PCB) archiválásra kerül vagy törölve lesz.

Állapotváltások

A folyamat állapotai közötti váltások az operációs rendszer ütemezője és egyéb rendszerműveletek eredményeként történnek. Például egy folyamat állapotot válthat futó állapotból várakozó állapotba, ha egy I/O műveletet kezdeményez, vagy kész állapotba kerülhet újra, ha az I/O művelet befejeződött.

Ezen állapotok megértése kulcsfontosságú az operációs rendszer folyamatkezelési stratégiáinak és az erőforrás-kezelési technikáinak tanulmányozásában. Ezek az állapotok segítenek az ütemezőnek a CPU idő hatékony elosztásában a különböző folyamatok között.

# Folyamat ütemezése

A folyamat ütemezése az operációs rendszerek egyik kulcsfontosságú funkciója, amely a CPU erőforrásainak elosztását irányítja a különböző folyamatok között. Az ütemező (scheduler) célja, hogy maximalizálja a CPU kihasználtságát, minimalizálja a várakozási időt, és biztosítsa a rendszer hatékony működését. Az ütemezési stratégiák több tényezőt is figyelembe vesznek, beleértve a folyamat prioritását, a felhasználói interaktív válaszidőt, és a rendszer általános teljesítményét. Itt van néhány alapvető folyamatütemezési algoritmus:

1. **First-Come, First-Served** (FCFS)

Az egyik legegyszerűbb ütemezési algoritmus, ahol a folyamatokat abban a sorrendben ütemezik, ahogy megérkeznek. Az FCFS hátránya, hogy hosszú CPU-blokkoló műveletek jelentősen megnövelhetik más folyamatok várakozási idejét (convoy effect).

2. **Shortest Job First** (SJF)

Ez az algoritmus a befejezéséhez szükséges legkevesebb időt igénylő folyamatot részesíti előnyben. Bár ez az algoritmus optimalizálja a várakozási időt, a valós végrehajtási idő előrejelzése gyakran nem lehetséges, és hosszú távú folyamatok éhenhalhatnak (starvation).

3. **Priority Scheduling**

A folyamatok prioritása alapján történik az ütemezés, ahol magasabb prioritású folyamatok kapnak előnyt. Azonban ez az algoritmus is hajlamos az éhenhalásra, különösen ha alacsony prioritású folyamatokról van szó.

**4. Round Robin (RR)**

Az egyik leggyakoribb preemptív ütemezési algoritmus, ahol minden folyamat egyenlő, rögzített hosszúságú időszeletet (time slice) kap a CPU-ból. Ez az algoritmus megfelelő válaszidőt biztosít a többfelhasználós rendszerekben, de a time slice méretének megfelelő beállítása kritikus.

**5. Multilevel Queue**

A folyamatokat különböző sorokba rendezik, gyakran prioritás vagy folyamat típusa szerint. Minden sor saját ütemezési algoritmussal rendelkezik, és a sorok közötti ütemezés prioritás alapján történik.

**6. Multilevel Feedback Queue**

Ez egy változata a multilevel queue-nak, ahol a folyamatok mozoghatnak a különböző sorok között a rendszerben eltöltött idejük vagy válaszuk alapján, lehetővé téve a rendszernek, hogy dinamikusan alkalmazkodjon a folyamat viselkedéséhez.

**7. Real-Time Scheduling**

Valós idejű operációs rendszerekben, ahol a feladatok időkritikusak, két fő típus létezik: kemény valós idejű (hard real-time), ahol a határidők betartása kötelező, és lágy valós idejű (soft real-time), ahol a határidők betartása kívánatos, de nem kötelező.

Az ütemezési stratégiák kiválasztása az operációs rendszer felhasználási módjától, a felhasználói igényektől,

## **d: Ütemezők**

Az ütemezők kulcsfontosságú komponensei az operációs rendszereknek, amelyek felelősek a processzor (CPU) idejének elosztásáért a különböző folyamatok és szálak között. A folyamatok ütemezése döntően befolyásolja a rendszer teljesítményét, válaszidejét és felhasználói élményét. A különböző típusú ütemezők eltérő stratégiákat alkalmaznak a folyamatok prioritásának, igényeinek és típusainak kezelésére.

1. Hosszú Távú Ütemező (Long-Term Scheduler) vagy Feladat Ütemező

A hosszú távú ütemező, más néven felvételi ütemező, dönti el, hogy mely folyamatokat vigye át az input sorból a memóriába a végrehajtás előkészítéséhez. Ez a típusú ütemező kevésbé gyakori a modern operációs rendszerekben, mert a folyamatok gyakran tartósan tartózkodnak a memóriában. A hosszú távú ütemező szabályozza a multiprogramozás mértékét, és dönt arról, hogy a rendszer hány folyamatot kezelhet egyszerre.

2. Középtávú Ütemező (Medium-Term Scheduler)

A középtávú ütemező, vagy más néven szwapoló, felelős a folyamatok memóriából való ki- és becseréléséért (swapping). Ez az ütemező optimalizálja a rendszer teljesítményét és a memória kihasználtságát azáltal, hogy szabályozza a memóriában egyidejűleg tárolt folyamatok számát.

3. Rövid Távú Ütemező (Short-Term Scheduler) vagy CPU Ütemező

A rövid távú ütemező, vagy CPU ütemező a leggyakoribb és legkritikusabb ütemező típus, amely dönti el, hogy melyik kész állapotban lévő folyamat kapja meg a CPU-t a következő végrehajtási ciklushoz. Ez az ütemező nagyon gyorsan működik, és gyakran váltogat a folyamatok között, hogy biztosítsa a rendszer interaktivitását és válaszképességét.

4. Valós Idejű Ütemező (Real-Time Scheduler)

Valós idejű ütemezők kemény és lágy valós idejű rendszerekben találhatók, ahol a feladatoknak meghatározott időkorlátokon belül kell befejeződniük. A valós idejű ütemezők garantálják, hogy a kritikus feladatok megfelelő prioritással és időben kerüljenek végrehajtásra, és gyakran alkalmaznak prioritás-alapú ütemezési algoritmusokat.

Ütemezési Stratégiák

Az ütemezők többféle stratégiát használhatnak, mint például:

Round Robin (RR): Minden folyamat egyenlő, korlátozott ideig tartó időszeletet kap a CPU használatára.

Priority-based: Folyamatok prioritása alapján történik az ütemezés, ahol magasabb prioritású folyamatok kapnak előnyt.

First-Come, First-Served (FCFS): A folyamatokat az érkezési sorrendjük alapján ütemezik.

Shortest Job First (SJF): A legkevesebb CPU időt igénylő folyamatok részesülnek előnyben.

Az ütemezők kialakítása és az általuk alkalmazott stratégiák nagymértékben befolyásolják az operációs rendszer teljesítményét, megbízhatóságát és felhasználói élményét, ezért ezek a rendszertervezés kulcselemei.

## Műveletek folyamatokon, folyamatok közötti kommunikáció

A folyamatokon végzett műveletek és a folyamatok közötti kommunikáció két alapvető aspektusa az operációs rendszerek működésének, amelyek lehetővé teszik a rendszer hatékony és biztonságos működését, valamint az alkalmazások közötti adatcserét.

Műveletek Folyamatokon

Folyamat Létrehozása (Creation): Új folyamatok létrehozása a rendszerben. Ez magában foglalhatja egy folyamat szülő általi létrehozását vagy egy teljesen új alkalmazás indítását. A folyamat létrehozása során az operációs rendszer kiosztja a szükséges erőforrásokat (pl. memória, fájl leírók) és inicializálja a folyamatkontroll blokkot (PCB).

Folyamat Megszüntetése (Termination): Egy folyamat befejezése vagy megszüntetése, amikor már nem szükséges, vagy ha hibát észlel a rendszer. A megszüntetés magában foglalja az erőforrások felszabadítását és a PCB törlését.

Folyamat Blokkolása és Feloldása (Blocking and Unblocking): Folyamatokat blokkolhatnak, ha várakozniuk kell egy adott eseményre (pl. I/O művelet befejeződése). Amint az esemény bekövetkezik, a folyamat feloldódik és ismét kész állapotba kerül.

Folyamat Várakoztatása és Folytatása (Suspending and Resuming): Egy folyamat várakoztatása, ha az erőforrásokat ideiglenesen más folyamatoknak kell átadni, vagy ha a folyamatot külső események miatt kell szüneteltetni. A várakoztatott folyamatokat később újra folytathatják.

Folyamatok Közötti Kommunikáció (IPC)

Üzenetküldés (Message Passing): Folyamatok közötti kommunikációra szolgáló mechanizmus, amelyben az üzeneteket a küldő folyamatból a fogadó folyamatba továbbítják az operációs rendszeren keresztül. Ez lehet szinkron vagy aszinkron, a folyamatok az üzenetküldés során blokkolódhatnak vagy nem.

Megosztott Memória (Shared Memory): Folyamatok egy előre meghatározott memória területen osztoznak, ahol adatokat helyezhetnek el és olvashatnak ki. Ez a módszer gyorsabb adatcserét tesz lehetővé, mivel közvetlen memória hozzáférést biztosít, de szükséges a megfelelő szinkronizáció (pl. szemaforokkal).

Csövek (Pipes): Egy speciális forma az üzenetküldésnek, amelyet gyakran használnak szekvenciális adatátvitelre. A csövek lehetnek névtelenek (anonim), amik csak egy szülő és annak gyermekfolyamatai között létesíthetők, vagy nevesítettek, amelyek tetszőleges folyamatok között hozhatók létre.

Szignálok (Signals): Egy alacsony szintű kommunikációs mechanizmus a folyamatok között, amelyek események vagy megszakítások jelzésére szolgálnak. Szignálok küldhetők a rendszer vagy más folyamatok által egy folyamat

## Távoli eljáráshívások

GPT

A távoli eljáráshívás (Remote Procedure Call, RPC) egy kommunikációs protokoll, amely lehetővé teszi egy program számára, hogy eljárásokat hívjon meg egy másik számítógépen (amely lehet távoli szerver vagy egy másik ügyfélkliens), mintha az a helyi gépen futna. Az RPC lényegében elrejti a hálózati kommunikáció bonyolultságait, lehetővé téve a fejlesztők számára, hogy egyszerűen és hatékonyan tervezhessenek és implementálhassanak elosztott alkalmazásokat.

Hogyan Működik az RPC

Eljárás Hívása:

A hívó folyamat egy eljárást hív meg, amely lokálisan tűnik, de valójában egy távoli szerveren fut. Az RPC keretrendszer az eljáráshívás paramétereit csomagolja (marshalling), átküldi a hálózaton keresztül, majd a szerveren kicsomagolja (unmarshalling).

Eljárás Végrehajtása:

A távoli szerver fogadja a hívást, kicsomagolja a paramétereket, és végrehajtja a kért eljárást, mintha helyi hívás lenne. A szerver ezt követően összecsomagolja a válaszadatokat (ha vannak), és visszaküldi azokat a hívónak.

Válasz Feldolgozása:

A hívó folyamat fogadja a választ, kicsomagolja, és folytatja a végrehajtást a válaszadatok felhasználásával.

RPC Jellemzők

Átláthatóság: Az RPC átláthatóságot biztosít, mivel a fejlesztők nem kell, hogy törődjenek a hálózati kommunikáció részleteivel; az eljáráshívások olyanok, mint a helyi hívások.

Nyelvfüggetlenség: Az RPC-t úgy tervezték, hogy támogassa a különböző programozási nyelveken írt klienseket és szervereket. Ezáltal az interfész definíciók nyelvfüggetlenek, lehetővé téve a különböző platformok közötti integrációt.

Szinkron és Aszinkron kommunikáció: Az RPC lehet szinkron, ahol a hívó blokkolódik a válasz megérkezéséig, vagy aszinkron, ahol a hívó folytatja a végrehajtást, és később dolgozza fel a választ.

Alkalmazási Területek

Elosztott Rendszerek: Az RPC ideális elosztott alkalmazásokban, ahol az adatok és szolgáltatások szétszórtak több gépen vagy szerveren.

Szolgáltatás-Orientált Architektúra (SOA): Az RPC-t gyakran használják webes szolgáltatások és SOA alkalmazásokban, ahol a szolgáltatások hálózaton keresztül érhetők el.

Az RPC technológia lehetővé teszi, hogy a szoftverfejlesztők hatékonyan kezeljék a hálózati kommunikációt, csökkentve a hálózati programozás bonyolultságát, miközben erős eszközt biztosít az elosztott rendszerek fejlesztéséhez.

## Üzenetátadás, csővezetékek

Az üzenetátadás (message passing) és a csővezetékek (pipes) két alapvető kommunikációs mechanizmus a folyamatok közötti adatcserére az operációs rendszerekben. Mindkettő lehetővé teszi az adatok és üzenetek átadását különböző folyamatok között, de különböző módokon és különböző szituációkban használhatók hatékonyan.

Üzenetátadás

Az üzenetátadás egy olyan módszer, amelyben a folyamatok explicit üzeneteket küldenek és fogadnak egymástól, általában az operációs rendszer által biztosított API-n keresztül. Ez lehet szinkron vagy aszinkron:

Szinkron üzenetátadás: Mindkét folyamat (küldő és fogadó) blokkolódik az üzenetküldés és -fogadás idejére. A küldő folyamat vár, amíg az üzenetet át nem veszik, és a fogadó vár, amíg az üzenet meg nem érkezik.

Aszinkron üzenetátadás: A küldő folyamat tovább folytathatja a végrehajtást az üzenet elküldése után, a fogadó folyamat pedig akkor dolgozza fel az üzenetet, amikor az elérhetővé válik.

Az üzenetátadás előnyei közé tartozik a folyamatok közötti erős elszigeteltség, ami növeli a rendszer biztonságát és stabilitását. Azonban az üzenetek kódolása és dekódolása (marshalling/unmarshalling) overheadet jelenthet, ami csökkentheti a kommunikáció hatékonyságát.

Csővezetékek (Pipes)

A csővezetékek olyan IPC mechanizmusok, amelyek lehetővé teszik a folyamatok számára, hogy egy "cső" mentén adatfolyamot küldjenek egymásnak. Két fő típusuk van:

Névtelen csövek (Anonymous pipes): Egyszerű kommunikációs csatorna, amely általában csak két, egymással rokon folyamat között használható. Támogatják csak az egysíkú adatátvitelt (az egyik irányba).

Nevesített csövek (Named pipes): Komplexebb kommunikációs csatorna, amely lehetővé teszi a kétirányú kommunikációt, és nem korlátozódik rokoni kapcsolatokra. A nevesített csövek lehetővé teszik különböző folyamatok közötti, akár különböző gépeken futó folyamatok közötti kommunikációt is.

A csővezetékek előnye a viszonylag egyszerű használat és a hatékony adatátvitel, különösen a streamolható adatok, mint például a szöveges logok vagy médiafájlok esetében. Azonban a csővezetékek nem biztosítanak ugyanolyan szintű elszigeteltséget és biztonságot, mint az üzenetátadás.

Alkalmazási területek

Üzenetátadás: Alkalmas összetett adatstruktúrák átadására és olyan alkalmazásokban, ahol fontos a biztonság és a folyamatok közötti adatelkülönítés.

Csővezetékek: Gyakran használják shell szkriptekben és batch folyamatokban, ahol az egyik program kimenetét közvetlenül egy másik program bemenetére kell átadni.

Mindkét kommunikációs mechanizmus fontos szerepet játszik az operációs rendszerekben, lehetővé téve a rugalmas és hatékony adatcserét a folyamatok között.