Operációs rendszerek

Folyamatok

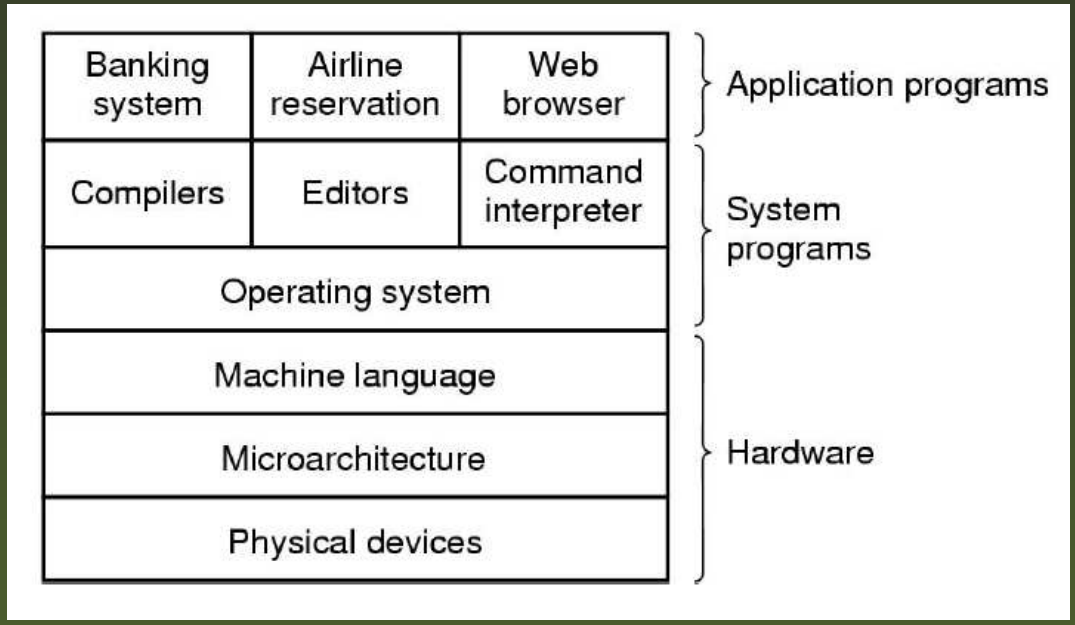
1.1

Pere László (pipas@linux.pte.hu)

PÉCSI TUDOMÁNYEGYETEM TERMÉSZETTUDOMÁNYI KAR
INFORMATIKA ÉS ÁLTALÁNOS TECHNIKA TANSZÉK

A rendszermag

Rendszermag (*kernel*): A szűk értelemben vett operációs rendszer, a program, amely a felhasználói programok és a hw közt álva az OS feladatát látja el.

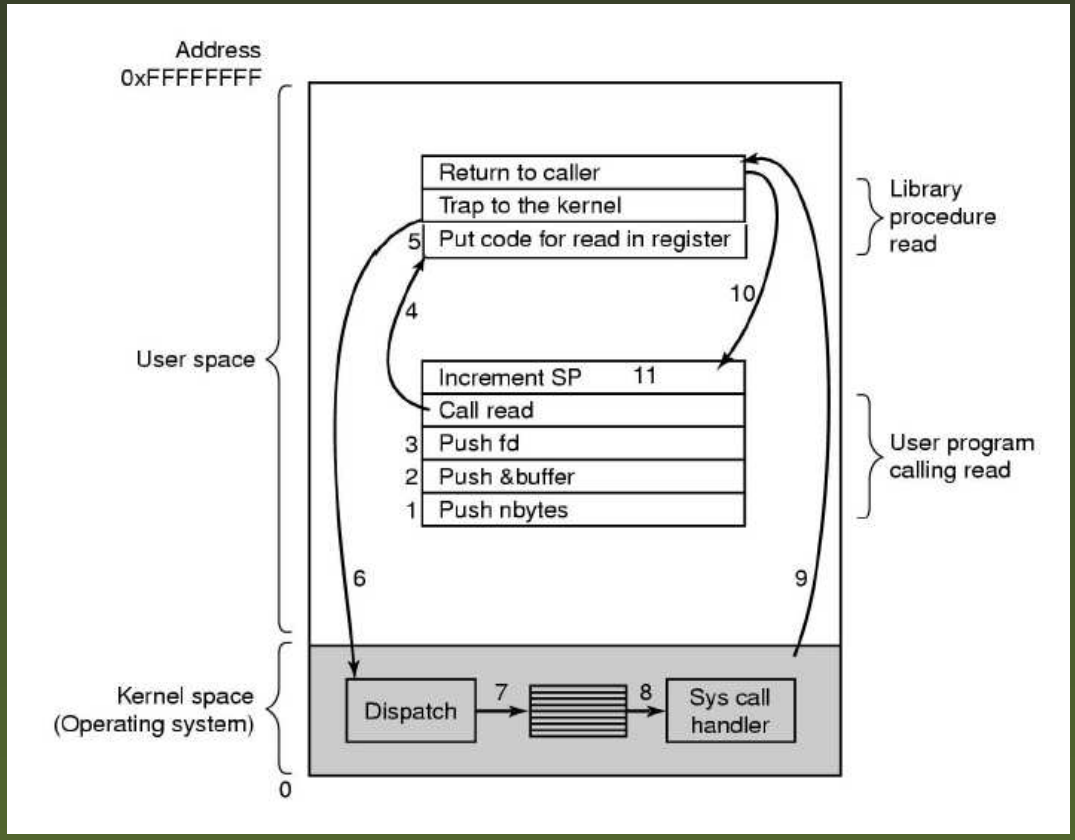


1. Ábra: *Az operációs rendszer helye*

A rendszermag futása közben a processzor privilegizált állapotban van.

A rendszerhívás

A rendszerhívás (*system call*) kérés a rendszermag felé, a folyamat, amelynek során a program a megfelelő módon kéri a hw vezérlését vagy más változtatást.



2. Ábra: *A rendszerhívás*

A folyamat

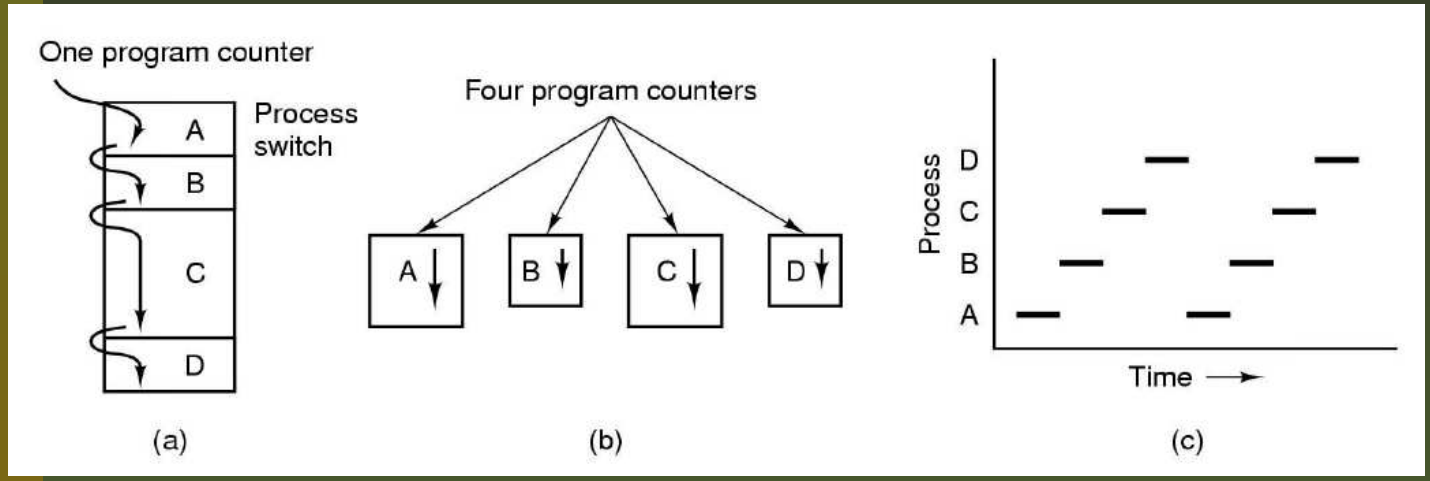
A folyamat (*process*) az operációs rendszerek elméletének alapfogalma, absztrakt fogalomként a futó programot jelöljük vele.

Figyelem! A folyamat nem program, ahogyan a főzés nem recept vagy a kirándulás nem utiterv.

A folyamat időben történő dinamikus esemény, a program pedig statikus leírás, amely megadja, hogy mikor mit kell tennünk.

Párhuzamosság

A számítógép látszólag egyszerre, párhuzamosan futtatja a folyamatokat, míg valójában egymás után végzi el az egyes folyamatok parancsait (*pseudoparallelism*).



3. Ábra: *A látszólagos párhuzamosság*

Időosztásos rendszerek

Olyan rendszerekben, ahol a processzor átkapcsolásokat végez a folyamatok közt, nem tehetünk előfeltételezéseket az időzítésre.

Multiprogramozott környezetben a folyamat lefutása általában nem megjósolható vagy megismételhető időzítés szerint történik.

A program nem feltételezheti, hogy az egyes műveletek milyen időzítés szerint történnek.

Folyamatok létrehozása

Folyamatokat a következő esetekben hozunk létre:

1. rendszerindítás (*system initialization*)
2. folyamatindító rendszerhívás (*system call*)
3. felhasználó indít folyamatot
4. kötegelt (*batch*) feladat indítása

Technikailag minden folyamat indításának oka rendszerhívás, rendszerindításkor azonban a kernel hívja a kernelt.

Folyamat befejezése

A folyamat a következő okok miatt fejeződhet be:

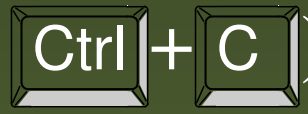
1. Normál kilépés rendszerhívással (tervezett).
2. Kilépés rendszerhívással, hiba miatt (tervezett).
3. Kilépés végzetes hiba miatt (nem tervezett).
4. Más folyamat általi megszakítás (nem tervezett).

A végzetes hibákat a folyamatok kezelhetik, általában elkerülhetik a megszakítást, ha saját hibakezelőt regisztrálnak.

A folyamat megszakításához a megfelelő jogkörrel kell rendelkezni.

Folyamathierarchia

Ha egy folyamat rendszerhívással egy újabb folyamatot hoz létre, speciális viszonyban – gyemek-szülő – maradnak. A gyermekfolyamat újabb folyamatokat hozhat létre, folyamathierarchiát építve.

Unix rendszerekben a folyamat és az összes leszármazottja folyamatcsoportot alkotnak. A folyamatcsoportok sorsa bizonyos mértékben közös (pl. jelzések fogadása, 

).

Windows rendszereken a handler átadható más folyamatnak, ami lerombolja a hierarchiát.

Folyamatindítás (UNIX)

A `fork()` másolatot készít a futó folyamat memóriaterületéről, gyermek szülő viszonyba kerülnek és ugyanarról a pontról folyik mindkettő végrehajtása.

Az `execve()` lecseréli a memóriatérképet (*memory image*) egy állományban található programra, amelyet a kezdőponttól indít.

A program úgy indít el másik programot, hogy egymás után hajtja végre a `fork()` majd az `execve()` rendszerhívásokat.

Folyamatindítás (Windows)

`CreateProcess()` rendszerhívás, 10 paraméterrel, további mintegy 100 függvény a folyamatok kezelésére. A Windows megoldás nagyon bonyolult.

Windows rendszereken a szülő és gyermek memóriaterülete az első pillanattól különböző. Úgy is mondhatnánk, hogy a Windows rendszerekből hiányzik a `fork()` rendszerhívás.

Az ütemező

Ha egy rendszeren több folyamat található, mint ahány processzor, az operációs rendszernek el kell döntenie, hogy mely időpillanatban mely folyamat fusson. Ezt a feladatot az operációs rendszer része, az ütemező (*scheduler*) látja el.

Az ütemező hely (több processzor) és időmultiplexeléssel osztja szét a processzort vagy processzorokat a folyamatok számára.

Folyamatok állapotai

A folyamatok a következő állapotokat vehetik fel:

- Fut (*running*). A processzor éppen a folyamatot alkotó utasításokat hajtja végre.

Folyamatok állapotai

A folyamatok a következő állapotokat vehetik fel:

- Fut (*running*). A processzor éppen a folyamatot alkotó utasításokat hajtja végre.
- Futásra kész (*ready*). A processzor azért nem a folyamat utasításait hajtja végre, mert más folyamattal van elfoglalva.

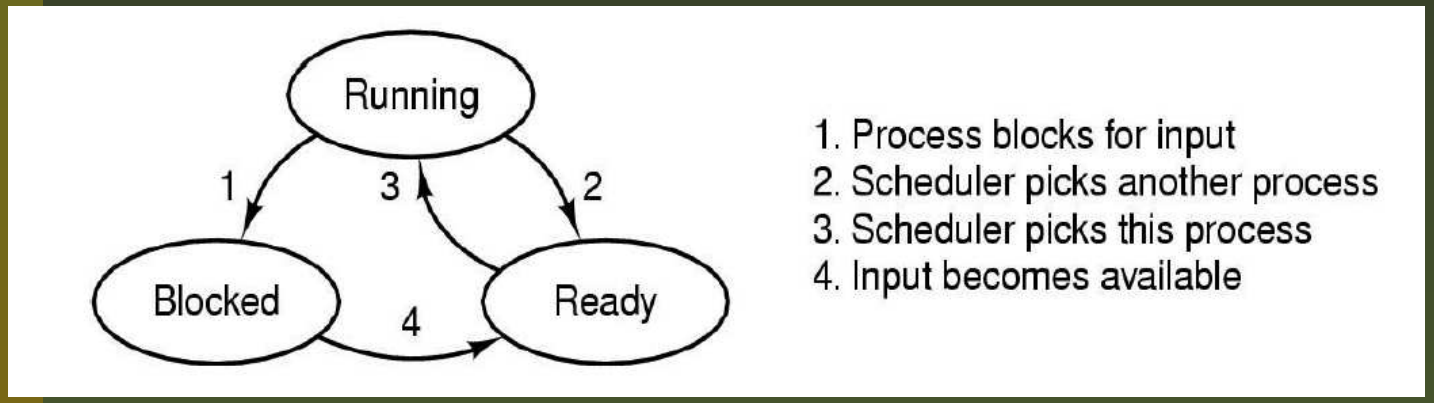
Folyamatok állapotai

A folyamatok a következő állapotokat vehetik fel:

- Fut (*running*). A processzor éppen a folyamatot alkotó utasításokat hajtja végre.
- Futásra kész (*ready*). A processzor azért nem a folyamat utasításait hajtja végre, mert más folyamattal van elfoglalva.
- Blokkolt (*blocked*). A folyamat külső eseményre vár, amelynek be kell következnie, hogy újra futásképes legyen.

Folyamatok állapotai

A következő ábra bemutatja, hogy mik azok az állapotváltozások, amelyeket a legtöbb operációs rendszer megvalósít:



4. Ábra: *A folyamatok állapotváltozásai*

Folyamatok állapotai

Az állpotátmenetek a következő események hatására következnek be:

1) A folyamat olyan rendszerhívást adott, amelynek perifériára kell várnia vagy a folyamat blokkolást kért.

Folyamatok állapotai

Az állpotátmenetek a következő események hatására következnek be:

1) A folyamat olyan rendszerhívást adott, amelynek perifériára kell várnia vagy a folyamat blokkolást kért.
2) Az ütemező úgy döntött, hogy épp elég régóta fut a folyamat, másnak adja a futás jogát (időmultiplexelés).

Folyamatok állapotai

Az állpotátmenetek a következő események hatására következnek be:

1) A folyamat olyan rendszerhívást adott, amelynek perifériára kell várnia vagy a folyamat blokkolást kért.
2) Az ütemező úgy döntött, hogy épp elég régóta fut a folyamat, másnak adja a futás jogát (időmultiplexelés).
3) Az ütemező futási jogot ad a folyamatnak.

Folyamatok állapotai

Az állpotátmenetek a következő események hatására következnek be:

1) A folyamat olyan rendszerhívást adott, amelynek perifériára kell várnia vagy a folyamat blokkolást kért.
2) Az ütemező úgy döntött, hogy épp elég régóta fut a folyamat, másnak adja a futás jogát (időmultiplexelés).
3) Az ütemező futási jogot ad a folyamatnak.
4) Bekövetkezett az esemény, amelyre a folyamatnak várnia kellett, a folyamat képessé vált a futásra.

Kötelező irodalom

1. ANDREW S. TANENBAUM: *Modern operating systems* second edition (2001), Prentice Hall, Upper Saddle River, New Jersey, ISBN0-13-031358-0, 71–80. o.