Čakajoči proces je postal spet pripravljen. Ali si morda že takoj zasluži CPE?

4. tekoči proces končal
   CPE je prosta. Kdo jo dobi naslednji?
5. prispel nov proces
   Prispel nov proces, ki je pripravljen. Ali si morda zasluži CPE?

Ločimo dva pristopa:

- razvrščanje povzročita **le 1 in 4.**

Proces bo torej izgubil CPE le, če je bil onemogočen oz. se je končal, torej *če ne more več teči.* V nasprotnem pa bo nemoteno tekel.
Zato to imenujemo **razvrščanje brez prekinjanja** (nonpreemtive sch.)
(Pozor: Prekinitve, ki so posledica običajnih prekinitvenih zahtev, so seveda možne, saj ne sprožajo razvrščanja, temveč le začasno izvajanje prekinitvenih servisnih programov PSP - ti pa 'ne štejejo'. Zato bi bil ustreznejši prevod **razvrščanje brez odvzemanja**, saj se procesu ne da odvzeti CPE, lahko pa se ga še vseeno prekine v standardnem smislu.)

- poleg 1 in 4 razvrščanje sproži **še vsaj ena od 2,3,5.**

(Za odvzem sedaj zadošča tudi iztek dodeljenega časa ali pa sprememba v vrsti pripravljenih procesov.) V tem primeru proces lahko izgubi CPE, četudi bi še lahko tekel.
Torej je lahko res *prekinjen.* To je **razvrščanje s prekinjanjem** (preemptive sch.)

7. **Naštej stanja procesa v CPU!**
   Stanja procesa so: *ustvarjen (nov), pripravljen(čaka, da se mu dodeli procesna enota), teče (se izvaja), čaka (ustavljen), končan.*

8. **Pojasni signale v UNIXu! Imenuj razlog, zakaj so signali lahko nezanesljivi.**
   Signal je programska prekinitev s strani sistema, drugega programa ali samega sebe. Signal je možno poslati, prestreči, prezreti ali obravnavati. Nosijo malo informacije – predvsem za javljanje nenavadnih okoliščin ali napak.
   Signali so lahko nezanesljivi, ker se lahko v izjemnih okoliščinah obnašajo drugače, kot je predvideno.

9. **Rezidentni monitor**
   Uporablja se za avtomatizacijo delovanja poslov. Ob koncu vsakega posla kliče naslednjega. Ob zagonu se sam naloži v pomnilnik ter ostane tam celi čas. Za delovanje potrebuje kontrolne kartice, zato tudi interpreter kartic, nalagalnik ter gonilnike za V/I naprave.

10. **Prekrivanje operacij**
    Realizirano je bili z ločenim V/I ali pa preko spoolinga. Ločeni V/I se je pojavil, ko se je pojavil magnetni trak, saj je trak bil vmesni člen men V/I in CPU. Tako so lahko hkrati potekalo V/I in računske operacije. Problem je bil, ker je trak imel zaporedni dostop.
    Spooling se je pojavil z diskom, saj imajo ti neposredni dostop, zato se uporabljajo kot vmensi mediji za začasno hranjenje V/I podatkov. Spooling uporablja branje vnaprej in tako omogoči, da je na disko lahko hkrati pripravljenih več poslov.

11. **Multiprogramska paketna obdelava**
    Smisel je, da ko se posel ustavi, OS izbere novega. S tem se pojavijo zahteve po razvrščanju poslov, po upravljanju s pomnilnikom, po dodeljevanju procesorja (razvrščanju na procesorju) ter zaščiti

1. **Razloži semafor!**
   Semafor je orodje za časovno usklajevanje procesov. Je celoštevilska spremenljivka S, nad katero se izvajajo operacije inicializacija, wait(S) ter signal(S) – signal = S+=1, wait pa čaka dokler je S<=0, nato S-=1. Ponavadi sta ti dve operaciji realizirani na OS.

2. **Razloži spooling!**
   Spooling je branje vnaprej – s tem omogoči, da je na disku več pripravljenih poslov hkrati. Omogoča, da se disk lahko uporabi kot velik vmesnik za začasno hranjenje vhodno/izhodnih podatkov. Omogoča prekrivanje V/I operacij ter računskih operacij.

3. **Razloži kritične odseke / sekcije / dele ! Katere kriterije bi morali upoštevati za splošno rešitev problema kritičnega dela?**
   Kritični odsek je del programa oz. procesa, ki dostopa do skupnega sredstva. Bistveni pogoj za rešitev tega problema je da je v kritičnem odseku v vsakem trenutku le en proces.
   Kriteriji: medsebojno izključevanje v vsakih okoliščinah, ne sme priti do zastoja, ne sme priti do nepredvideno dolgega odlaganja, izvajanje enega procesa znotraj kritičnega dela ne sme ovirati napredovanja drugih procesov izven kritičnega dela.

4. **Naštej razlike med vtičnicami in cevmi!**
   Cevi so enosmerni kanali – za komunikacijo sta potrebni dve, na vtičnice pa se priklopijo dvosmerni kanali. Cevi (razen FIFO, t.j. poimenovane cevi) lahko obstajajo le med dvema procesoma v sorodstvu (otrok-otrok, roditelj-otrok), za vtičnice pa ta relacija ni potrebna.

5. **Socketi**
   Vtičnice so enostaven programski vmesnik za komunikacijo med procesi na istem ali različnih računalnikih. Pred pričetkom komunikacije morata oba procesa P in Q ustvariti vsaj svojo vričnico. Nato pove vsak svoji vtičnici kakšen naslov bo imela (v mreži je to IP). Sedaj se lahko pogovarjata – strežnik (listen, accept), odjemalec (connect), za oba (send, recv, read, write, close).

6. **Razloži razvrščanje – prekinitveno in neprekinitveno!**
   Ločimo tri vrste razvrščanja – na procesorju (kratkoročno – teče, pripravljen), razvrščanje poslov (dolgoročno - nov) in menjavanje (srednjeročno - pripravljen).
   Razvrščanje opravlja razvrščevalnik, ki je del OS.
   Na enoprocesorskem sistemu v nekem trenutku teče kvečjemu en proces. Ostali procesi bodisi *čakajo* (na nek dogodek) ali pa so *pripravljeni* (čakajo na CPE). Med pripravljenimi se slednjič enega *izbere* in sedaj se njemu *dodeli* CPE, temu se reče **preklop**. Pravimo, da je procesor zamenjal okolje svojega delovanja. Tak preklop imenujemo tudi menjava okolja (context switch). Pri menjavi okolja (preklopu) sta v resnici udeležena dva programa OS:
   razvrščevalnik (scheduler), ki izbere enega od pripravljenih procesov,
   dodeljevalnik (dispatcher), ki izbranemu procesu v resnici dodeli procesor, tj. izvede menjavo okolja.

   V katerih okoliščinah je možno/nujno pognati razvrščevalnik?
   Potem, ko je:
   1. tekoči proces postal onemogočen
      Proces ne more nadaljevati, zato bo šel med čakajoče in sprostil CPE. Kdo dobi naslednji CPE?
   2. tekoči proces je prekinila ura
      Tekoči proces je rabil svojo časovno rezino. Kdo dobi naslednjo čas. rezino? On ali kdo drug?
   3. čakajoči proces dočakal izpolnitev pogojev za nadaljevanje

Podobnosti: tudi nit je lahko v stanjih (nova, prekinjena, teče, čaka ,končana), tudi nit lahk oustvari nove niti (hčere) ter jih konča in uniči, na enoprocesorskem sistemu tudi niti tekmujejo za procesor.
Razlike: komunikacija med istorodnimi nitmi je hitrejša kot med procesi, istorodne niti niso zaščitene ena pred drugo tako kot procesi, ustvarjanje / končanje niti je hitrejše kot pri procesih.

25. **IPC**
Je del OS, ki skrbi za komunikacijo in sinhronizacijo med procesi.

12. **Sistem z dodeljevanje časa – večopravilni OS**
Multiprogramski sistem paketne obdelave, ki uporablja še timer. Pri njem se pojavijo zahteve po uporavljanju z zunanjim pomnilnikom, navideznim pomnilnikom, zahteve po datotečnem sistemu, po zaščiti, po dodeljevanju procesorja ter po sinhronizaciji in komunikaciji.

13. **Krmilnik**
Nadzoruje ustrezno napravo ali več naprav.

14. **Bootstrap**
Majhen, enostaven program, ki inicializira računalniški sistem (registre CPE, pomnilnike, krmilnike...), naloži jedro OS v pomnilnik in mu nato preda nadzor.

15. **Programska prekinitev**
Sistemski klic, s katerim tekoči program zahteva pomoč OS – trap, exception.

16. **Pooling – programsko izpraševanje**
Pri prekinitvah. Ko se pojavi prekinitev, CPE prekine delo, zažene PSP ki ugotovi kdo je zahteval prekinitev, zažene ustrezni PSP in nato nadaljuje z delom.

17. **Vektorsko prekinjanje**
Ko naprave zahteva prekinitev, v CPE pošlje tudi indeks i, s pomočjo katerega nato CPE v prenitvenem vektorju (tabeli v pomnilniku) poišče ustrezen PSP ter ga zažene. Ostalo je normalno.

18. **Vhodno izhodni sistem**
Sestavljen je iz V/I naprav, krmilnikov ter programov za V/I operacije. Krmilnik skrbi za podrobnosti pri prenosu podatkov – na njih se izvajajo V/I programi.

19. **Sinhroni V/I**
Proces, ki je zahteval V/I operacijo, čaka da se ta konča. Lahko izvede kako zanko, ukaz wait ali pa sam spremlja statusni register.

20. **Asinhroni V/I**
Proces, ki je zahteval V/I operacijo, nadaljuje svoje izvajanje. Če ne more nadaljevati dela, ker čaka na podatke, sam sebe blokira. Je primeren za počasne naprave.

21. **DMA krnilnik**
Se uporablja pri hitrih napravah za prenos med pomnilnikom in V/I napravami, brez posredovanja CPU.

22. **Sistemski klic**
Z njegovo pomočjo programer zaprosi OS, da ta v njegovem imenu izvede neko (zanj prepovedano) akcijo. Nekateri višji programski jeziki pa te klice (read, load, store, create, open, execute ...) vseeno omogočajo.

23. **Nit**
Je tok izvajanja programa do danega trenutka. Če je sistem enoprocesorski, procesor posveti vsaki niti nekaj časa, tako da vse napredujejo. Če pa je sistem večprocesorski, pa se lahko vsaka nit izvaja na svojem procesorju (to ni paralelno, ampak sočasno).

24. **Podobnosti in razlike med nitmi in procesi**