**PB153 - OS a jejich rozhraní**

10 otázek (max. 20b. -> na E potřeba aspoň 10b.) času víc než dost a kdyby někdo přesto nestíhal, tak by ho to prý nechal dopsat.

**1.** Proces typicky přechází ze stavu běžící (running) do čekající (waiting)

při: **(slide 7. str. 5)**

**\*a) synchronní I/O**

b) asynchronní I/O

c) ukončení procesu

d) odstavení od procesoru algoritmem RR - (Round Robin)

e) žádná z uvedených možností

**1.1.** Proces typicky přechází ze stavu běžící (running) do připravený

(ready) při:

a) synchronní I/O

b) asynchronní I/O

c) ukončení procesu

**\*d) plánování pomocí RR (odstavení od procesoru algoritmem RR)**

e) žádná z uvedených možností

+ další kombinace (new, ready, waiting, running, terminated)

**2.** **Preemptivní SJF - s přeskakováním**

**!!!P1 - 7s, P2 - 4s, P3 - 1s, P4 - 4s!!!**

**P1,P2,P3,P2,P4,P1 -> 0-2,2-4,4-5,5-7,7-11,11-16**

Který proces poběží v čase 10? **(Slide 7. str. 13)**

a) P1

b) P2

c) P3

**\*d) P4**

e) žádná z uvedených možností

**Nepreemptivní SJF - bez přeskakování**

**P1,P3,P2,P4 -> 0-7,7-8,8-12,12-16**

**3.** Co znamená RES? **(Slide 11. str. 5 (obr.))**

**\*a) rezidentní paměť, která nebyla od swapovaná**

**(rezidentní čase, neodswapovaná fyzická paměť, který proces použil)**

b) rezidentní paměť, která nemůže být od swapovaná

c) resistentní paměť ...

d) ...

e) ...

**4.** V "page table" nebylo nic uvedeno a my jsme to měli

doplnit (výsledek úplně stejný jako je ve slidě). Která z možností

page/frame/valid je správně? **(Slide 11. str. 13)**

**(Správně: 0/4/v, 1/x/i, 2/6/v, 3/x/i, 4/x/i, 5/9/v, 6/x/i, 7/x/i)**

a) 0/5/v, 1/7/i ...

b) 0/4/i, 1/9/v ...

c) 0/4/v, 1/x/i ...

**\*d) žádná z uvedených možností**

**5.** OS s mikrojádrem **(Slide 4. str. 19-20)**

a) je rychlejší, protože je menši režie uvnitř jádra

**\*b) je stabilnější a bezpečnější**

**\*c) snadná přenositelnost OS**

d) příkladem je MS DOS

e) ...

**(Mikrojádro: malé jádro plnící nezbytně málo funkcí, primitivní správa**

**paměti, komunikace mezi procesy (IPC), některé funkce se přesouvá z**

**jádra do "uživatelské oblasti", ovladače HW, služby sestému**

**souborů, virtualizace paměti, komunikace procesů předáváním zpráv**

**Výhody: snadná přenositelnost (malé jádro), vyšší spolehlivost, bezpečnost,**

**flexibilita, služby jsou poskytovány jednotně (výměnou zpráv)**

**Nevýhody: zvýšená režie, volání služeb je nahrazeno výměnou zpráv mezi**

**procesy)**

**6.** Systémové volání

**\*a) příkladem jsou LINUXová volání open, close, exit**

b) příkladem jsou LINUXová volání openFile, closeFile, ...

**\*c) slouží jako rozhraní mezi aplikací a jádrem OS**

d) LINUX má asi 5000 volání (jádra v2.2 -> 200 a v2.6 -> 300volání)

**\*e) lze je implementovat pomocí přerušení**

**6.1.** Předávání argumentů (parametrů) při systémových volání je realizované

**(Slide 3. str. 24)**

**\*a) registry**

**\*b) zásobník**

**\*c) pointer na strukturu s datama uložený v paměti patřičného procesu**

**(blok v paměti & pointeru)**

d) IN/OUT instrukce

e) kritické sekce

**7.** SYSENTER: **(Slide 3. str. 29)**

a) slouží k pomalému systémovému volání

**\*b) slouží k rychlému systémovému volání**

c) žádná z odpovědí

**8.** fork(): **(Slide 5. str. 22)**

**\*a) vytvori novy proces v LINUXu**

b) vytvori novy proces LINUXu jako kopii rodice a nasledne rodice zabije **?\*c) jeho ekvivalent ve WIN API je createProcess()**

d)

e)

**9.** vlákna: **(slide 6.)**

**\*a) přepnutí vlákna při plánování CPU je rychlejší než přepnutí procesu**

**\*b) každé vlákno si udržuje svůj zásobník**

c) vlákna spolu komunikují přes IPC

**\*d) aktualní verze LINUXu podporuje kernel-level threads**

e) ...

**10.** (RAG bez cyklu). Výše uvedený obrázek je: **(Slide 9. str. 9)**

a) RAG s cyklem, tedy došlo k uváznutí

**\*b) RAG bez cyklu, tedy nedošlo k uváznutí**

c) Wait-for graf s cyklem, tedy došlo k uváznutí

d) RAG bez cyklu, tedy došlo k uváznutí

e) Wait-for graf bez cyklu, tedy nedošlo k uváznutí

**10.1.** Další příklady RAG grafů

**P1->R1->P2->R3->P3 ==> bez cyklu bez uváznutí**

**P1->R1->P2->R3->P3->R2.1->P1 && R2.2->P2 ==> s cyklem s uváznutím**

**P1->R1.1->P2&&R1.2->P3->R2.1->P1&&R2.2->P4 ==> s cyklem bez uváznutí**

**P1->R1->P2->R2->P3->R3->P1 ==> s cyklem s uváznutím**

**11.** IPC **(Slide 8. str. 31)**

**\*a) semafor**

**\*b) sdílená paměť**

**\*c) roury**

d) registry

e)

**(Správné odpovědi: signály, roury, zprávy, semafory, sdílená paměť)**

**12.** Digitální podpis ovladačů jádra **(Slide 4. str. 37-38)**

**\*a) uplatňuje se pro zvýšení bezpečnosti**

**\*b) je ve win 7**

c) něco s DOSem a jeho použitím

d) je nahrazen jiným (typem)

e) ...

**13.** Paměť pro vyrovnávání bloků I/O **(Slide 2 str. 9)**

**\*a) buffer**

b) cache

c) best-fit

d) wait - for cyklus

**14.** Plánovací rozhodnutí může vydat v okamžiku, kdy proces

**(Slide 7. str. 6) <- plánování CPU**

a) přechází ze stavu běžící (running) do stavu čekající (waiting)

b) přechází ze stavu běžící (running) do stavu připravený (ready)

c) přechází ze stavu čekající (waiting) do stavu připravený (ready)

d) končí

**A a D se označují jako nepreemptivní plánování (plánování bez předbíhání)**

**B a C se označují jako preemptivní plánování (plánování s předbíháním)**