Architektura a koncepce OS – OS a HW (archOS_HW)

Aby fungoval OS s preemptivním multitaskingem, musí HW obsahovat:
1. (+2) přerušovací systém (interrupt system)
2. (+2) časovač

Při používání DMA:
1. (+1) se přenosu neúčastní CPU
2. (+1) je nutné alokovat od systému kanál (DMA)
3. (+2) se urychlí činnost systému, protože se nepoužívá CPU
4. (+2) je obvyklé používat také přerušovací systém

Architektura a koncepce OS – Jádro OS (archOS_kernel)

Která funkce by měla být povolena pouze v režimu kernel?
1. (+2) povolení přerušení
2. (+2) změna kořenového adresáře (chroot)
3. (+2) zákaz přerušení
4. (+2) nastavení času systémových hodin
5. (+2) zachycení a obsluha interruptu
6. (+2) ovládání V/V zařízení

TRAP:
1. (+2) je skok z režimu user do režimu kernel
2. (+2) se používá pro systémová volání

Systémové volání:
1. (+1) slouží procesům ke zpřístupnění funkcí OS
2. (+1) slouží procesům k ovládání V/V zařízení

Architektura a koncepce OS – Typy OS (archOS_typy)

Mezi distribuované systémy patří:
1. (+1) Beowulf cluster
2. (+1) ParallelKnoppix

Architektura a koncepce OS – Funkce OS (archOS_fce)

Hlavní funkce OS jsou:
1. (+1) správa prostředků
2. (+1) abstrakce a rozšíření počítače
3. (+1) management zdrojů
4. (+1) virtualizace a rozšíření HW

Timesharing je:
1. (+1) způsob multiprogrammingu
2. (+1) sdílení (dělení) času CPU mezi procesy uživatelů OS

Pod pojmem spooling rozumíme v oblasti OS také:
1. (+1) techniku ukládání úloh do fronty pro dávkové systémy
2. (+1) odkládání dat pro pomalejší V/V zařízení

Multiprogramingem můžeme označit:
1. (+1) způsob práce plánovače OS
2. (+1) (pseudo)paralelní běh více úloh
3. (+1) jeden ze způsobů práce plánovače OS
4. (+1) (pseudo)současný běh více procesů

Správa paměti – Pojmy o paměti (mem_teorie)

Položka stránkové tabulky obsahuje:
1. (+1) číslo rámce
2. (+1) řídicí bity

Položka segmentové tabulky neobsahuje:
1. (+1) číslo segmentu
4. (+1) offset od bázové adresy

Vnější fragmentace paměti:
1. (+1) je odstraněna použitím stránkování

Segmentace:
1. (+1) usnadňuje sdílení paměti mezi procesy
2. (+1) pomáhá implicitně řešit problém ochrany

Stránkování paměti:
1. (+1) odstraňuje vnější fragmentaci
2. (+1) je pro programátora transparentní

Thrashing:
1. (+1) je neefektivní využití CPU při neustálé výměně paměťových stránek
2. (+1) může být způsobován odkládáním paměti na disk, když je tato část za okamžik potřebná

Specifické OS – Systémy reálného času (otherOS_RT)

Mezi typické vlastnosti RTOS patří:
1. (+1) rychlé přepínání kontextu
2. (+1) multitasking

Mezi typické vlastnosti RTOS nepatří:
1. (+1) nepreemptivní plánování
2. (+1) plánování zaměřené na maximální využití CPU

Specifické OS – Vestavěné systémy

Podíl trhu mikročipů mimo vestavěné systémy je zhruba:
1. (+1) < 5 % (2%)

Podíl trhu mikročipů pro vestavěné systémy je zhruba

4. (+1) > 90 % (98%)

Sdílení prostředků – Kritická sekce (sdileni_KS)

Vstup do kritické sekce lze dostatečně ošetřit pomocí:
1. (+2) prostředků OS, pomocí semaforu
2. (+2) prostředků OS, pomocí předávání zpráv
1. (+2) prostředků jazyka C# nebo Java, pomocí monitoru
2. (+2) prostředků posixových vláken, pomocí binárního semaforu

Výhodou řešení vstupu do kritické sekce pomocí zákazu přerušení je:
1. (+1) jednoduchost použití
2. (+1) neaktivní čekání

Nevýhodou řešení kritické sekce pomocí zákazu přerušení je:
1. (+1) nemožnost použití na SMP-systémech
2. (+1) zvyšování latence systému

Řešení vstupu do kritické sekce pomocí předávání zpráv jako prostředku OS:
1. (+1) používá krátkou vstupní a výstupní sekci
2. (+1) je výhodné pro používání neaktivního čekání

Zbytková sekce je:
1. (+2) část kódu procesu(ů)

Výhodou řešení vstupu do kritické sekce pomocí instrukce typu test-and-set je:
1. (+1) možnost použití na SMP-systémech
2. (+1) jednoduchost použití

Monitor jako prostředek ošetření vstupu do kritické sekce je:
1. (+1) nástroj programovacího jazyka

Monitor jako prostředek ošetření vstupu do kritické sekce:
1. (−2) je nevhodný, protože používá aktivní čekání
2. (−2) je nevhodný, protože příliš zvyšuje latenci systému
3. (−2) nelze použít
4. (−2) se běžně používá v jazyce C, C++ a Delphi
5. (+2) žádná z výše uvedených možností

Sdílení prostředků – Synchronizace (sdileni_sync)

Synchronizování procesů tak, aby od bariéry běžely oba současně, lze dosáhnout dostatečně pomocí:
1. (+2) prostředků OS, pomocí předávání zpráv

Sdílení prostředků – Semafory (sdileni_sem)

Semafor v OS neobsahuje:
1. (−1) čítač (čítací proměnnou)
2. (−1) funkci signal (up)
3. (−1) funkci wait (down)
4. (−1) frontu (proměnnou pro seznam procesů)

5. (+1) žádná z výše uvedených možností

Semafor v OS obsahuje:
1. (+1) čítač (čítací proměnnou)
2. (+1) funkci signal (up)
3. (+1) funkci wait (down)
4. (+1) frontu (proměnnou pro seznam procesů)

Procesy – Plánování (proc_plan)

Hlavní cíle plánování procesů jsou:
1. (+1) spravedlnost
2. (+1) rovnováha zatížení subsystémů

Hlavní cíle plánování procesů na real-timeových systémech jsou:
1. (+1) prediktabilita (předvídatelnost)
2. (+1) dodržení (časových) termínů

Hlavní cíle plánování procesů na dávkových systémech jsou:
1. (+1) minimalizace obratu (turnaround time)
2. (+1) maximální zátěž (využití) CPU

Hlavní cíle plánování procesů na interaktivních systémech jsou:
2. (+1) nízká latence a odezva
4. (+1) proporcionalita (přiměřenost) k očekávání uživatelů

Procesy – Stavy procesů (proc_stavy)

Třístavový model procesu zahrnuje následující stav:
1. (+1) blokovaný
2. (+1) běžící
3. (+1) připravený

Sedmistavový model procesu nezahrnuje následující stavy:
1. (+1) odložený, spustitelný, spící
2. (+1) vyčerpaný, naplánovaný, odblokovaný

Sedmistavový model procesu zahrnuje (mj.) následující stavy:
1. (+1) blokovaný odložený, běžící, ukončený
2. (+1) odložený blokovaný, blokovaný, připravený
3. (+1) běžící, blokovaný, nový
4. (+1) připravený, běžící, ukončený

Procesy – Komunikace procesů (proc_kom)

Vyberte správné tvrzení o rourách:
1. (+1) slouží ke komunikaci procesů
2. (+1) v posixových systémech se s nimi pracuje obdobně jako se soubory

Vyberte správné tvrzení o socketech:
1. (+1) slouží ke komunikaci procesů

2. (+1) v posixových systémech se s nimi pracuje obdobně jako se soubory

Procesy – Vlákna

Vlákna sdílejí se zbytkem procesu:
1. (+1) paměť

Můžou taky sdílet
přidělené prostředky (např. otevřené soubory)

Vlákna nesdílejí se zbytkem procesu (s ostatními vlákny):
2. (+1) zásobník
3. (+1) stav (kontext)

Vlákna nesdílejí se zbytkem procesu (s ostatními vlákny) :
1. (+1) zásobník
2. (+1) stav (kontext)

Nevýhodou implementace vláken bez podpory OS je:
1. (+1) page-fault způsobí zastavení ostatních vláken
3. (+1) nutnost převést blokovaná volání na neblokovaná

Bezpečnost OS (security)

Simulování přihlašovací obrazovky se nazývá:
1. (+1) login spoofing

UNIX používá standadně pro uložení hesel funkci crypt() založenou na algoritmu DES. Jak dlouho řádově trvá vypočítání původního hesla z uloženého záznamu hesla na běžném PC:
1. (+1) nelze

UNIX používá standadně pro uložení hesel funkci crypt() založenou na algoritmu DES. Jak dlouho řádově trvá uhodnutí řádně voleného funkčního hesla na běžném PC, pokud máme k dispozici uložený záznam.
3. (+2) tisíce až desetitisíce let

Mezi nejčastější útoky na systém patří:
1. (+1) využití chyby ve službách typu buffer overflow
2. (+1) hádání uživatelských loginů a jejich hesel slovníkovou metodou

Různé

Kolik definuje sysvinit standardně tzv. runlevelů na Linuxu?
1. (2) 7
Který runlevel sysvinit je v Linuxu standardně určen pro shutdown?
1. (2) 0
Který runlevel sysvinit je v Linuxu standardně určen pro správu v jednouzivatelském rezimu?
1. (1) 1

Procesy

Procesy – Přepínání kontextu (proc_ctxSw)

Postup:

1. spočítáme si kolikrát ve sledovaném čase (50 ms) byl součet časové kvantum (11ms) + context-switch (2ms) -> (11 + 2) + (11 + 2) + (11 + 2) -> 3x
2. context-switch (2ms) * počet výskytů (3 * 2) -> 6 ms
3. čas promrhaný všemi context-switchi za sledovaný čas vydělíme sledovaným časem 6 / 50 * 100 = 12%

Kolik procent času CPU je promrháno během 50 ms, pokud context-switch zabere 2 ms a časové kvantum bude 11 ms a právě bylo přepnuto na proces:
(+2) 12 %
(-2) 18 %
(-2) 82 %
(-2) 88 %
(-2) žádná z výše uvedených možností

.Kolik procent času CPU je promrháno během 57 ms, pokud context-switch zabere 3 ms a časové kvantum bude 9 ms a právě bylo přepnuto na proces:
(+2) 21 % = (4 * 3) / 57 * 100
(-2) 25 %
(-2) 75 %
(-2) 79 %
(-2) žádná z výše uvedených možností

.Kolik procent času CPU je promrháno během 60 ms, pokud context-switch zabere 3 ms a časové kvantum bude 9 ms a právě bylo přepnuto na proces:
(−2) 21 %
(+2) 25 %
(−2) 75 %
(−2) 79 %
(−2) žádná z výše uvedených možností

.Kolik procent času CPU je promrháno během 158 ms, pokud context-switch zabere 2 ms a časové kvantum bude 38 ms a právě bylo přepnuto na proces:
(+2) < 4 %
(-2) 5 %
(-2) 95 %
(-2) > 96 %
(-2) žádná z výše uvedených možností

.Kolik procent času CPU je promrháno během 170 ms, pokud context-switch zabere 4 ms a časové kvantum bude 25 ms a právě bylo přepnuto na proces:
(+2) < 12 %
(−2) 16 %
(−2) 84 %
(−2) > 88 %
(−2) žádná z výše uvedených možností

Procesy – Využití procesoru (proc_CPUutil)

Postup:

1. jak dlouho čekají (1/3) umocníme počtem procesů (1/3)*(1/3)*(1/3) nebo (1/3)^3 = 1/27
2. nesčítá se to protože procesy běží současně(paralelně) a ne za sebou(seriově)

.Počítač má paměť pro současný běh 3 procesů. Tyto procesy čekají průměrně třetinu času na dokončení V/V operace. Kolik průměrně času je procesor (CPU) nevyužit?
(-2) 1/3
(-2) 1/9
(+2) 1/27
(-2) 2/9
(-2) žádná z výše uvedených možností

.Počítač má paměť pro současný běh 3 procesů. Tyto procesy dvě třetiny času čekají na dokončení V/V operace. Kolik průměrně času je procesor (CPU) nevyužit?
(-2) 2/3
(-2) 1/2
(+2) 8/27 (2/3)^3
(-2) 4/9
(-2) žádná z výše uvedených možností

.Počítač má paměť pro současný běh 3 procesů. Tyto procesy polovinu času čekají na dokončení V/V operace. Kolik průměrně času je procesor (CPU) nevyužit?
(-2) 1/2
(+2) 1/8
(-2) 1/16
(-2) 1/4
(-2) žádná z výše uvedených možností

Počítač má paměť pro současný běh 4 procesů. Tyto procesy polovinu času čekají na dokončení V/V operace. Kolik průměrně času je procesor (CPU) nevyužit?
(-2) 1/2
(-2) 1/8
(+2) 1/16
(-2) 1/4
(-2) žádná z výše uvedených možností

Souborové systémy (FS)

Dávat si pozor:

- jestli jde o B nebo o KB
- sektor má 512 B, pokud není dáno jinak

Postup:

1. alokační blok (4 sektory) * standardní velikost sektoru (512 B) = 2048 B = 2 kB
2. všechny velikosti souborů vydělíme velikostí alokačního bloku (2 kB)
3. 104 / 2 = 52 bloků
4. 194 B = 0,194 / 2 = 0,097 (zaokrouhlíme na 1 blok) -> mrháme
5. 310 B = 0,310 kB / 2 = 0,155 (zaokrouhlíme na 1 blok) -> mrháme
6. u 1. souboru je použito celých 52 bloků beze zbytku a nemrháme
7. u 2. souboru je použit 1 blok a mrháme (1 * 2 - 0,194 = 1,806 kB) // 1*2 = blok * velikost alokačního bloku
8. u 3. souboru je použit 1 blok a mrháme (1 * 2 - 0,310 = 1,69 kB)
9. celkově použitá paměť 2 kB * 54 bloků = 108 kB
10. celkově promrháno (1,806 + 1,69) = 3,496
11. procent promrháno: (100* 3,496)/108 = 3.237037 => zaokrouhleno na 4%

Souborové systémy – Alokační bloky na FS (FS_cluster)

.Kolik procent místa je přibližně promrháno, pokud se na souborový systém s alokačním blokem 4 sektory uloží 3 soubory o velikostech 104 kB, 194 B a 310 B?

(−1) 91 %
(−1) 9 %
(+1) 4 %
(−1) 96 %
(−1) žádná z výše uvedených možností

.Kolik procent místa je přibližně promrháno, pokud se na filesystém s alokačním blokem 8 sektorů uloží 3 soubory o velikostech 8 B, 17 kB a 250 B?

(+1) 40 %
(−1) 98 %
(−1) 2 %
(−1) 46 %
(−1) žádná z výše uvedených možností

.Kolik procent místa je přibližně promrháno, pokud se na filesystém s alokačním blokem 16 kB uloží 3 soubory o velikostech 50 kB, 18 kB a 10 B?

(+1) 40 %
(-1) 50 %
(-1) 60 %
(-1) 30 %
(-1) žádná z výše uvedených možností

.Kolik procent místa je přibližně promrháno, pokud se na filesystém s alokačním blokem 16 kB uloží 3 soubory o velikostech 51 kB, 18 B a 17 kB?

(+1) 40 %

(-1) 50 %
(-1) 60 %
(-1) 30 %
(-1) žádná z výše uvedených možností

.Kolik procent místa je přibližně promrháno, pokud se na souborový systém s alokačním blokem 16 sektorů uloží 3 soubory o velikostech 54 kB, 256 B a 453 B?
(−1) 97 %
(−1) 3 %
(+1) 25 %
(−1) 75 %
(−1) žádná z výše uvedených možností

.Kolik procent místa je přibližně promrháno, pokud se na filesystém s alokačním blokem 16 sektorů uloží 3 soubory o velikostech 60 kB, 18 kB a 5 B?
(+1) 19 %
(−1) 9 %
(−1) 22 %
(−1) 30 %
(−1) žádná z výše uvedených možností

.Kolik procent místa je přibližně promrháno, pokud se na souborový systém s alokačním blokem 16 sektorů uloží 3 soubory o velikostech 90 kB, 225 B a 321 B?
(−1) 98 %
(−1) 2 %
(+1) 20 %
(−1) 80 %
(−1) žádná z výše uvedených možností

.Kolik procent místa je přibližně promrháno, pokud se na souborový systém s alokačním blokem 16 sektorů uloží 3 soubory o velikostech 54 kB, 256 B a 453 B?
(−1) 97 %
(−1) 3 %
(+1) 25 %
(−1) 75 %
(−1) žádná z výše uvedených možností

.Kolik procent místa je přibližně promrháno, pokud se na souborový systém s alokačním blokem 64 sektorů uloží 3 soubory o velikostech 68 kB, 148 B a 535 B?
1. (−1) 98 %
2. (−1) 1 %
3. (+1) 58 %
4. (−1) 43 %
5. (−1) žádná z výše uvedených možností

.Kolik procent místa je přibližně promrháno, pokud se na souborový systém s alokačním blokem 64 sektorů uloží 3 soubory o velikostech 105 kB, 152 B a 309 B?
1. (−1) téměř 100 %
2. (−1) skoro 0 %
3. (+1) 46 %
4. (−1) 54 %
5. (−1) žádná z výše uvedených možností

Souborové systémy – FAT (velikost souborového systému) (FS_FATFSs)

Postup:

1. umocníme 2 na FAT-číslo -> 2^12 = 4096 adres bloků
2. počet sektorů alokační jednotky (bloku nebo clusteru) * standardní velikost sektoru

   = 8 sektorů * 512 B = 4096 B = 4 kB

3. počet adres * velikost alokačního bloku = 4096 * 4 kB = 16384 kB = 16 MB

.Při velikosti clusteru (alokační jednotky) 8 sektorů je maximální velikost filesystému FAT12:
(−2) 8 MB
(+2) 16 MB
(−2) 32 MB
(−2) 64 GB
(−2) žádná z výše uvedených možností

.Při velikosti clusteru (alokační jednotky) 8 sektorů je maximální velikost souborového systému FAT12:
(−2) 4 MB
(−2) 8 MB
(+2) 16 MB
(−2) 32 MB
(−2) žádná z výše uvedených možností

.Při velikosti clusteru (alokační jednotky) 8 sektorů je maximální velikost filesystému FAT16:
(−2) 128 MB
(+2) 256 MB
(−2) 512 MB
(−2) 1 GB
(−2) žádná z výše uvedených možností

.Při velikosti clusteru (alokační jednotky) 16 sektorů je maximální velikost filesystému FAT16:
(-2) 128 MB
(-2) 256 MB
(+2) 512 MB
(-2) 1 GB
(-2) žádná z výše uvedených možností

.Při velikosti clusteru (alokační jednotky) 64 sektorů je maximální velikost souborového systému FAT12:
(−2) 32 MB
(−2) 64 MB

(+2) 128 MB
(−2) 256 MB
(−2) žádná z výše uvedených možností

Souborové systémy – FAT (velikost tabulky) (FS_FATs)

Pozor na:

- velikost adresy !!! FAT16 = 16 bitů = 2 B, FAT32 = 32 bitů = 4 B
- velikost sektoru pokud není zadána tak je 512 B

Postup:

1. velikost FS v kB = 180 * 1024 = 184320 kB
2. velikost alokačního bloku (clusteru nebo jednotky) = počet sektorů * velikost jednoho sektoru = 32 * 512 B = 16384 B = 16 kB
3. počet adres = velikost FS / velikost al. bloku = 11520 adres
4. jelikož FAT12 může mít pouze 2^12 = 4096 adres **není** toto možné!!! (11520 > 4096)

Jaká bude velikost tabulky FAT12 při velikosti clusteru (alokační jednotky) 32 sektorů a velikosti souborového systému 180 MB:
(−2) 32 kB
(−2) 16 kB
(−2) 8 kB
(−2) 4 kB
(+2) žádná z výše uvedených možností

Postup:

1. velikost FS v kB = 560 * 1024 = 573440 kB
2. velikost alokačního bloku (clusteru nebo jednotky) = počet sektorů * velikost jednoho sektoru = 4 * 512 B = 2048 B = 2 kB
3. počet adres = velikost FS / velikost al. bloku = 286720 adres
4. jelikož FAT16 může mít pouze 2^16 = 65536 adres **není** toto možné!!! (286720 > 65536)

Jaká bude velikost tabulky FAT16 při velikosti clusteru (alokační jednotky) 4 sektory a velikosti souborového systému 560 MB:
(−2) 1120 kB
(−2) 560 kB
(−2) 280 kB
(−2) 140 kB
(+2) žádná z výše uvedených možností

Postup:

1. velikost FS v kB = 2 * 1024 * 1024 = 2097152 kB
2. velikost alokačního bloku (clusteru nebo jednotky) = počet sektorů * velikost jednoho sektoru = 8 * 512 B = 4096 B = 4 kB
3. počet adres = velikost FS / velikost al. bloku = 524288 adres
4. jelikož FAT16 může mít pouze 2^16 = 65536 adres **není** toto možné!!! (524288 > 65536)

.Jaká bude velikost tabulky FAT16 při velikosti clusteru (alokační jednotky) 8 sektorů a velikosti filesystému 2 GB:
(−2) 2 MB
(−2) 1 MB
(−2) 512 kB
(−2) 128 kB
(+2) žádná z výše uvedených možností

.Jaká bude velikost tabulky FAT16 při velikosti clusteru (alokační jednotky) 16 sektorů a velikosti filesystému 400 MB:
(−2) 200 kB
(+2) 100 kB
(−2) 50 kB
(−2) 25 kB
(−2) žádná z výše uvedených možností

počet adres je víc než může být (116480 > 65536)
.Jaká bude velikost tabulky FAT16 při velikosti clusteru (alokační jednotky) 16 sektorů a velikosti souborového systému 910 MB:
(−2) 454 kB
(−2) 200 kB
(−2) 113 kB
(−2) 56 kB
(+2) žádná z výše uvedených možností

Postup:

1. velikost FS v kB = 480 * 1024 = 491520 kB
2. velikost alokačního bloku (clusteru nebo jednotky) = počet sektorů * velikost jednoho sektoru = 64 * 512 B = 32768 B = 32 kB
3. počet adres = velikost FS / velikost al. bloku = 15360 adres
4. velikost tabulky = velikost jedné adresy * počet adres = (16(fat číslo) / 8(bitů na Bajt)) * 15360 = 30720 B = 30 kB

.Jaká bude velikost tabulky FAT16 při velikosti clusteru (alokační jednotky) 64 sektorů a velikosti souborového systému 480 MB:
(−2) 60 kB
(+2) 30 kB
(−2) 15 kB
(−2) 7 kB
(−2) žádná z výše uvedených možností

.Jaká bude velikost tabulky FAT32 při velikosti clusteru (alokační jednotky) 4 sektory a velikosti filesystému 32 GB:
(−2) 32 MB
(−2) 16 MB
(−2) 8 MB
(−2) 4 MB
(+2) žádná z výše uvedených možností (výsledek má být 64 MB)

.Jaká bude velikost tabulky FAT32 při velikosti clusteru (alokační jednotky) 4 sektory a velikosti filesystému 100 GB:
(−2) 400 MB
(+2) 200 MB
(−2) 50 MB
(−2) 25 MB
(−2) žádná z výše uvedených možností

.Jaká bude velikost tabulky FAT32 při velikosti clusteru (alokační jednotky) 16 sektorů a velikosti souborového systému 1000 GB:
(−2) 1000 MB
(+2) 500 MB
(−2) 125 MB
(−2) 62 MB
(−2) žádná z výše uvedených možností

Správa paměti

Dávat si pozor na:

- nezapomenout odčítat
- best-fit jede vždycky od začátku a přidá tam, kde toho nejméně zbyde
- (Exact-or) Worst fit (Buď stejný a nebo nejvíc volný blok)
- Když má blok velikost 0 tak vypadne a na jeho pozici se dostane následující

Správa paměti – Metody alokace (pořadí bloky) (mem_MAbloky)

Best-fit

.V paměti jsou volné bloky o velikostech 11 kB, 4 kB, 21 kB, 17 kB a 7 kB. Které bloky jsou vybrány pro postupnou alokaci: 12 kB, 10 kB a 8 kB, použije-li se algoritmus best-fit?
(−2) 3., 1., 3.
(−2) 3., 4., 1.
(+2) 4., 1., 3.
(−2) 3., 1., 4.
(−2) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 17 kB, 23 kB, 29 kB, 4 kB a 10 kB. Jaké bude pořadí vybraných bloků při postupné alokaci 5 kB, 13 kB a 11 kB, použije-li se algoritmus best-fit?
(−2) 1., 2., 1.
(+2) 5., 1., 2.
(−2) 1., 2., 3.
(−2) 3., 3., 3.

(−2) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 23 kB, 30 kB, 4 kB, 10 kB a 17 kB. Jaké bude pořadí vybraných bloků při postupné alokaci 13 kB, 12 kB a 9 kB, použije-li se algoritmus best-fit?
1. (−2) 1., 2., 1.
2. (+2) 5., 1., 4.
3. (−2) 1., 2., 2.
4. (−2) 2., 1., 2.
5. (−2) žádná z výše uvedených možností

V paměti jsou volné bloky o velikostech 26 kB, 32 kB, 6 kB, 13 kB a 19 kB. Jaké bude pořadí vybraných bloků při postupné alokaci 15 kB, 13 kB a 11 kB, použije-li se algoritmus best-fit?
1. (−2) 1., 2., 1.
2. (+2) 5., 4., 1.
3. (−2) 1., 2., 2.
4. (−2) 2., 4., 1
5. (−2) žádná z výše uvedených možností

(Exact-or) Worst fit (Buď stejný a nebo nejvíc volný blok)
.V paměti jsou volné bloky o velikostech 11 kB, 4 kB, 21 kB, 17 kB a 7 kB. Které bloky jsou vybrány pro postupnou alokaci: 12 kB, 10 kB a 8 kB, použije-li se algoritmus (exact- or) worst-fit?
(−2) 3., 1., 3.
(+2) 3., 4., 1.
(−2) 4., 1., 3.
(−2) 3., 1., 4.
(−2) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 19 kB, 26 kB, 32 kB, 6 kB a 13 kB. Jaké bude pořadí vybraných bloků při postupné alokaci 6 kB, 15 kB a 13 kB, použije-li se algoritmus (exact-or-)worst-fit?
(−2) 1., 2., 1.
(−2) 4., 1., 4.
(−2) 1., 2., 3.
(+2) 4., 3., 4.
(−2) žádná z výše uvedených možností

First-fit *První z řady. Projíždíš pro každé číslo celou řadu a hledáš první hodnotu do které se vejde*
.V paměti jsou volné bloky o velikostech 8 kB, 15 kB, 27 kB, 33 kB a 8 kB. Jaké bude pořadí vybraných bloků při postupné alokaci 10 kB, 8 kB a 6 kB, použije-li se algoritmus first-fit?
(+2) 2., 1., 2.
(−2) 2., 1., 4.
(−2) 2., 3., 3.
(−2) 4., 1., 2.
(−2) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 11 kB, 4 kB, 21 kB, 17 kB a 7 kB. Které bloky jsou vybrány pro postupnou alokaci: 12 kB, 10 kB a 8 kB, použije-li se algoritmus first-fit?
(+2) 3., 1., 3.
(−2) 3., 4., 1.

(−2) 4., 1., 3.
(−2) 3., 1., 4.
(−2) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 16 kB, 22 kB, 29 kB, 3 kB a 9 kB. Jaké bude pořadí vybraných bloků při postupné alokaci 10 kB, 8 kB a 5 kB, použije-li se algoritmus first-fit?
1. (+2) 1., 2., 1.
2. (−2) 1., 5., 1.
3. (−2) 1., 2., 2.
4. (−2) 3., 2., 3.
5. (−2) žádná z výše uvedených možností

Next-fit

.V paměti jsou volné bloky o velikostech 11 kB, 4 kB, 21 kB, 17 kB a 7 kB. Které bloky jsou vybrány pro postupnou alokaci: 12 kB, 10 kB a 8 kB, použije-li se algoritmus next-fit?
(−2) 3., 1., 3.
(+2) 3., 4., 1.
(−2) 4., 1., 3.
(−2) 3., 1., 4.
(−2) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 21 kB, 28 kB, 2 kB, 8 kB a 15 kB. Jaké bude pořadí vybraných bloků při postupné alokaci 8 kB, 15 kB a 13 kB, použije-li se algoritmus next-fit?
a. (−2) 1., 2., 1.
b. (−2) 4., 4., 1.
c. (+2) 1., 2., 2.
d. (−2) 4., 4., 2.
e. (−2) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 22 kB, 29 kB, 3 kB, 9 kB a 16 kB. Jaké bude pořadí vybraných bloků při postupné alokaci 13 kB, 11 kB a 9 kB, použije-li se algoritmus next-fit?
(−2) 1., 2., 1.
(−2) 5., 1., 4.
(+2) 1., 2., 2.
(−2) 2., 1., 4.
(−2) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 26 kB, 32 kB, 6 kB, 13 kB a 19 kB. Jaké bude pořadí vybraných bloků při postupné alokaci 15 kB, 13 kB a 11 kB, použije-li se algoritmus next-fit?
(−2) 1., 2., 1.
(−2) 5., 4., 1.
(+2) 1., 2., 2.
(−2) 2., 4., 1
(−2) žádná z výše uvedených možností

Správa paměti – Metody alokace (velikost bloků) (mem_MAvelBl)

Best-fit

.V paměti jsou volné bloky o velikostech 11 kB, 4 kB, 21 kB, 17 kB a 7 kB. Jak velké budou volné bloky po postupné alokaci 12 kB, 10 kB a 8 kB, použije-li se algoritmus best-fit?
(-3) 1 kB, 4 kB, 1 kB, 17 kB a 7 kB
(-3) 3 kB, 4 kB, 9 kB, 7 kB a 7 kB
(+3) 1 kB, 4 kB, 13 kB, 5 kB a 7 kB

(-3) 1 kB, 4 kB, 9 kB, 9 kB a 7 kB
(-3) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 11 kB, 17 kB, 21 kB, 14 kB a 7 kB. Jak velké budou volné bloky po postupné alokaci 12 kB, 10 kB a 7 kB, použije-li se algoritmus best-fit?
(−3) 1 kB, 5 kB, 14 kB, 14 kB a 7 kB
(−3) 11 kB, 5 kB, 3 kB, 14 kB a 7 kB
(+3) 1 kB, 17 kB, 21 kB a 2 kB
(−3) 1 kB, 21 kB, 14 kB a 7 kB
(−3) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 19 kB, 26 kB, 32 kB, 6 kB a 13 kB. Jak velké budou volné bloky po postupné alokaci 6 kB, 15 kB a 13 kB, použije-li se algoritmus best-fit?
(−3) 11 kB, 32 kB, 6 kB a 13 kB
(+3) 4 kB, 26 kB a 32 kB
(−3) 13 kB, 11 kB, 19 kB, 6 kB a 13 kB
(−3) 19 kB, 26 kB a 17 kB
(−3) žádná z výše uvedených možností

(Exact-or) Worst fit
.V paměti jsou volné bloky o velikostech 11 kB, 4 kB, 21 kB, 17 kB a 7 kB. Jak velké budou volné bloky po postupné alokaci 12 kB, 10 kB a 8 kB, použije-li se algoritmus (exact-or)worst-fit?
(-3) 1 kB, 4 kB, 1 kB, 17 kB a 7 kB
(+3) 3 kB, 4 kB, 9 kB, 7 kB a 7 kB
(-3) 1 kB, 4 kB, 13 kB, 5 kB a 7 kB
(-3) 1 kB, 4 kB, 9 kB, 9 kB a 7 kB
(-3) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 16 kB, 22 kB, 28 kB, 3 kB a 9 kB. Jak velké budou volné bloky po postupné alokaci 10 kB, 8 kB a 5 kB, použije-li se algoritmus (exact-or-)worst-fit?
1. (−3) 1 kB, 14 kB, 28 kB, 3 kB a 9 kB
2. (−3) 1 kB, 22 kB, 28 kB, 3 kB a 1 kB
3. (−3) 6 kB, 9 kB, 28 kB, 3 kB a 9 kB
4. (+3) 16 kB, 14 kB, 13 kB, 3 kB a 9 kB
5. (−3) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 17 kB, 23 kB, 29 kB, 4 kB a 10 kB. Jak velké budou volné bloky po postupné alokaci 5 kB, 13 kB a 11 kB, použije-li se algoritmus (exact-or-)worst-fit?
(−3) 1 kB, 10 kB, 29 kB, 4 kB a 10 kB
(−3) 4 kB, 12 kB, 29 kB, 4 kB a 5 kB
(−3) 12 kB, 10 kB, 18 kB, 4 kB a 10 kB
(+3) 17 kB, 23 kB, 4 kB a 10 kB
(−3) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 18 kB, 24 kB, 30 kB, 4 kB a 11 kB. Jak velké budou volné bloky po postupné alokaci 10 kB, 9 kB a 6 kB, použije-li se algoritmus (exact-or-)worst-fit?
(−3) 2 kB, 15 kB, 30 kB, 4 kB a 11 kB
(−3) 3 kB, 24 kB, 30 kB, 4 kB a 1 kB

(−3) 8 kB, 9 kB, 30 kB, 4 kB a 11 kB
(+3) 18 kB, 15 kB, 14 kB, 4 kB a 11 kB
(−3) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 21 kB, 28 kB, 2 kB, 8 kB a 15 kB. Jak velké budou volné bloky po postupné alokaci 8 kB, 15 kB a 13 kB, použije-li se algoritmus (exact-or)worst-fit?
1. (−3) 13 kB, 13 kB, 2 kB, 8 kB a 2 kB
2. (−3) 8 kB, 28 kB a 2 kB
3. (−3) 13 kB, 2 kB, 8 kB a 15 kB
4. (+3) 21 kB, 15 kB a 2 kB
5. (−3) žádná z výše uvedených možností

First-fit

.V paměti jsou volné bloky o velikostech 11 kB, 4 kB, 21 kB, 17 kB a 7 kB. Jak velké budou volné bloky po postupnoé alokaci 12 kB, 10 kB a 8 kB, použije-li se algoritmus first-fit?
(+3) 1 kB, 4 kB, 1 kB, 17 kB a 7 kB
(−3) 3 kB, 4 kB, 9 kB, 7 kB a 7 kB
(−3) 1 kB, 4 kB, 13 kB, 5 kB a 7 kB
(−3) 1 kB, 4 kB, 9 kB, 9 kB a 7 kB
(−3) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 22 kB, 29 kB, 3 kB, 9 kB a 16 kB. Jak velké budou volné bloky po postupné alokaci 13 kB, 11 kB a 9 kB, použije-li se algoritmus first-fit?
(+3) 18 kB, 3 kB, 9 kB a 16 kB
(−3) 11 kB, 29 kB, 3 kB a 3 kB
(−3) 9 kB, 9 kB, 3 kB, 9 kB a 16 kB
(−3) 11 kB, 16 kB, 3 kB a 16 kB
(−3) žádná z výše uvedených možností

Next-fit

.V paměti jsou volné bloky o velikostech 8 kB, 15 kB, 27 kB, 33 kB a 8 kB. Jak velké budou volné bloky po postupné alokaci 10 kB, 8 kB a 6 kB, použije-li se algoritmus next-fit?
a. (−3) 5 kB, 21 kB, 33 kB a 8 kB
b. (−3) 5 kB, 27 kB, 33 kB a 2 kB
c. (−3) 8 kB, 5 kB, 33 kB a 8 kB
d. (−3) 15 kB, 21 kB, 23 kB a 8 kB
e. (+3) žádná z výše uvedených možností (má být 8 kB, 5 kB, 13 kB, 33 kB a 8 kB)

.V paměti jsou volné bloky o velikostech 11 kB, 4 kB, 21 kB, 17 kB a 7 kB. Jak velké budou volné bloky po postupnoé alokaci 12 kB, 10 kB a 8 kB, použije-li se algoritmus next-fit?
(−3) 1 kB, 4 kB, 1 kB, 17 kB a 7 kB
(+3) 3 kB, 4 kB, 9 kB, 7 kB a 7 kB
(−3) 1 kB, 4 kB, 13 kB, 5 kB a 7 kB
(−3) 1 kB, 4 kB, 9 kB, 9 kB a 7 kB
(−3) žádná z výše uvedených možností

.V paměti jsou volné bloky o velikostech 17 kB, 23 kB, 30 kB, 4 kB a 10 kB. Jak velké budou volné bloky po postupné alokaci 11 kB, 8 kB a 6 kB, použije-li se algoritmus next-fit?
(−3) 15 kB, 30 kB, 4 kB a 10 kB
(−3) 23 kB, 30 kB, 4 kB a 2 kB
(+3) 6 kB, 9 kB, 30 kB, 4 kB a 10 kB

(−3) 17 kB, 15 kB, 13 kB, 4 kB a 10 kB
(−3) žádná z výše uvedených možností

V paměti jsou volné bloky o velikostech 23 kB, 29 kB, 4 kB, 10 kB a 16 kB. Jak velké budou volné bloky po postupné alokaci 14 kB, 12 kB a 9 kB, použije-li se algoritmus next-fit?
(−3) 17 kB, 4 kB, 10 kB a 16 kB
(−3) 11 kB, 29 kB, 4 kB, 1 kB a 2 kB
(+3) 9 kB, 8 kB, 4 kB, 10 kB a 16 kB
(−3) 11 kB, 15 kB, 4 kB, 10 kB a 7 kB
(−3) žádná z výše uvedených možností

V paměti jsou volné bloky o velikostech 23 kB, 30 kB, 4 kB, 10 kB a 17 kB. Jak velké budou volné bloky po postupné alokaci 13 kB, 12 kB a 9 kB, použije-li se algoritmus next-fit?
a. (−3) 1 kB, 18 kB, 4 kB, 10 kB a 17 kB
b. (−3) 11 kB, 30 kB, 4 kB, 1 kB a 4 kB
c. (+3) 10 kB, 9 kB, 4 kB, 10 kB a 17 kB
d. (−3) 11 kB, 8 kB, 4 kB, 10 kB a 17 kB
e. (−3) žádná z výše uvedených možnost

Správa paměti – Převod adres (mem_addrConv)

Pozor na:

- pořadí záznamu - pokud nám vyjde 0x1 -> 2 záznam, 0x0 -> 1. záznam atd.
- 4 bity jsou použity na jednu hodnotu v HEX soustavě (F = 1111)

Na segmenty (virtuální - sčítání)

Postup:

1. 16bitu = 2^4 bitu => offset jsou 4 hodnoty
2. 0x12012, takze po utrzeni offsetu(2012) zůstává 0x1 to znamená 2.pozice v tabulce což je 0xD5348 0x7FFFFF
3. k záznamu 0xD5348 se přičte offset D5348 + 2012 = D735A
4. takže 0xD735A je spravna odpoved

Pokud proces je rozdělen na 3 segmenty, offset v adrese je 16bitový a segmentová tabulka obsahuje (mj.) polozky:

base	***limit***
0x014DB	**0x00FFFF**
0xD5348	**0x7FFFFF**
0x1AC01	**0x0FFFFF**

Lineární adresa proměnné s virtuální (logickou) adresou (v procesu) 0x12012 je:
1. (-3) 0x014DB2974
2. (-3) 0xD53482012
3. (-3) 0xD5348012
4. (+3) 0xD735A
5. (-3) žádná z výše uvedených možnost

Postup:

1. 16bitu = 2^4 bitu => offset 4
2. 0x2012, takze po utrzeni offsetu(2012) zůstává 0x to znamená 1.pozice v tabulce, což je 0x014DB 0x00FFFF
3. k 1. záznamu se přičte offset = 14DB + 2012 = 34ED a navíc je mimo oblast prvního segmentu

Pokud proces je rozdělen na 3 segmenty, offset v adrese je 16bitový a segmentová tabulka obsahuje (mj.) položky:

base limit
0x014DB 0x00FFFF
0xD5348 0x7FFFFF
0x1AC01 0x0FFFFF

Lineární adresa proměnné s virtuální (logickou) adresou (v procesu) 0x2012 je:
(-3) 0x014DB2012
(-3) 0xD53482012
(-3) 0xD5348012
(-3) 0xD735A
(+3) žádná z výše uvedených možností

Pokud proces je rozdělen na 4 segmenty, offset v adrese je 24bitový a segmentová tabulka obsahuje (mj.) položky:

base limit
0x014DB 0x00FFFF
0xD5348 0x7FFFFF
0x1AC01 0x0FFFFF
0x51BA8 0x007FFF

Lineární adresa proměnné s virtuální (logickou) adresou (v procesu) 0x1001010 je:
(−3) 0x014DB1010
(−3) 0xD53481010
(−3) 0xD5348001010
(−3) 0x10D6358
(+3) žádná z výše uvedených možností (vyšlo mi 0xD6358)

Pokud proces je rozdělen na 4 segmenty, offset v adrese je 24bitový a segmentová tabulka obsahuje (mj.) položky:

base limit
0x28DF969 0x0FFFFF
0x49F1273 0x7FFFFF
0x201E810 0x07FFFF
0x12F175A 0x000FFF

Lineární adresa proměnné s virtuální (logickou) adresou (v procesu) 0x100342 je:
1. (−3) 0x28DF969100342
2. (−3) 0x49F1273100342
3. (−3) 0x201E810100342
4. (−3) 0x29DFCAB (tohle by normálně vyšlo)
5. (+3) žádná z výše uvedených možností (ale nevejde se do limitu)

Pokud proces je rozdělen na 4 segmenty, offset v adrese je 28bitový a segmentová tabulka obsahuje (mj.) položky:

base limit

0xC20A31 0x0FFFFF
0x5BCCCB 0x0FFFFF
0x64ABB75 0x0FFFFF
0x1FEAD5F 0x00FFFF
Lineární adresa proměnné s virtuální (logickou) adresou (v procesu) 0x21403423 je:
(−3) 0x78AEF98
(−3) 0x64ABB751403423
(−3) 0x5BCCCB1403423
(−3) 0x64ABB751403423
(+3) žádná z výše uvedených možností (3. segment je od 0x64ABB75 do 0x74ABB74, takže vypočítaná adresa 0x78AEF98 je mimo)

Pokud proces je rozdělen na 6 segmentů, offset v adrese je 20bitový a segmentová tabulka obsahuje (mj.) položky:
base limit
0x705457 0x00FFF
0x2C08361 0x00FFF
0x3470BCB 0x00FFF
0x65B6785 0x0FFFF
0x16FD33F 0x000FF
0x4842EF9 0x00FFF
Lineární adresa proměnné s virtuální (logickou) adresou (v procesu) 0x411004 je:
(−3) 0x170E343
(−3) 0x16FD33F11004
(−3) 0x2C0836111004
(−3) 0x3470BCB11004
(+3) žádná z výše uvedených možností (5. segment je od 0x16FD33F do 16FD43E, takže vypočítaná adresa 0x170E343 je mimo)

Na stránky (lineární - přidává se na konec)

Postup:

1. 4 kB = 4096 B = 2^12
2. 12/4 = 3 znaky je offset (ABC) a 0x4 je řídící bit -> pátý frame (0x5B06)
3. přípíšeme offset nakonec frame => 0x5B06ABC

Pokud proces je rozdělen na 5 stránek velikosti 4 kB a stránková tabulka obsahuje (mj.) položky:
frame
0x303C
0x1583
0x1ABC
0xABC5
0x5B06
Fyzická adresa proměnné s lineární (logickou) adresou (v procesu) 0x4ABC je:
a. (+3) 0x5B06ABC
b. (−3) 0x5B064ABC
c. (−3) 0x65C2
d. (−3) 0xA5C2
e. (−3) žádná z výše uvedených možností

Postup:

1. 4 kB = 4096 B = 2^12 -> stránka má 12 bitů
2. 12 / 4 = 3 znaky v HEX soustavě
3. 0x25A0 odřízneme zprava a dostaneme offset 5A0 a řídící bit 0x2 (3. rámec)
4. ke 3. rámci **0x1C23** připíšeme offset 5A0 => 0x1C235A0

Pokud proces je rozdělen na 3 stránky velikosti 4 kB a stránkovací tabulka obsahuje (mj.) položky:
frame
0x80A3
0x60A3
0x1C23
Fyzická adresa proměnné s lineární (logickou) adresou (v procesu) 0x25A0 je:
(+3) 0x1C235A0
(−3) 0x21C3
(−3) 0x41C3
(−3) 0x8643
(−3) žádná z výše uvedených možností

Postup:

1. 64 kB = 65536 B = 2^16
2. 16 / 4(4 bity na jednu hodnotu v HEX soustavě) = 4 -> odebereme z lin. adresy 0x108FC zprava 4 hodnoty (08FC) => 0x1 - 2. záznam (**0x2153)**
3. na konec **0x2153** přidáme 08FC -> 0x215308FC

Pokud proces je rozdělen na 12 stránek velikosti 64 kB a stránková tabulka obsahuje (mj.) položky:
frame
0xAAE4
0x2153
0xD2C1
0x4692
0x34C3
0xBAD0
0xBED3
0x1243
0x680F
0xA467
0xED56
0x41B4

Fyzická adresa proměnné s lineární (logickou) adresou (v procesu) 0x108FC je:
(−3) 0x21538FC
(+3) 0x215308FC
(−3) 0xED568FC

(−3) 0xED56108FC
(−3) žádná z výše uvedených možností

Architektura a koncepce OS (archOS)

Architektura a koncepce OS – Funkce OS (archOS_fce)

Hlavní funkce OS jsou:
(+1) správa prostředků
(+1) abstrakce a rozšíření počítače
(−1) grafické uživatelské rozhraní
(−1) prioritní řazení procesů
(−1) žádná z výše uvedených možností

Hlavní funkce OS jsou:
(+1) management zdrojů
(+1) virtualizace a rozšíření HW
(−1) GUI
(−1) nepreemptivní plánování procesů
(−1) žádná z výše uvedených možností

Multiprogramingem můžeme označit:
(−1) programování v týmu
(−1) programování aplikací pro audio a video
(+1) způsob práce plánovače OS
(+1) (pseudo)paralelní běh více úloh
(−1) žádná z výše uvedených možností

Multiprogrammingem můžeme označit:
(-1) programování více programátory
(-1) programování multimediálních aplikací
(+1) jeden ze způsobů práce plánovače OS
(+1) (pseudo)současný běh více procesů
(-1) žádná z výše uvedených možností

Pod pojmem spooling rozumíme v oblasti OS také:
(+1) techniku ukládání úloh do fronty pro dávkové systémy
(+1) odkládání dat pro pomalejší V/V zařízení
(−1) algoritmus přidělování paměti vláknům
(−1) sdílení paměti mezi V/V zařízeními
(−1) žádná z výše uvedených možností

Timesharing je:
(+1) způsob multiprogrammingu
(+1) sdílení (dělení) času CPU mezi procesy uživatelů OS
(−1) úspora času při kopírování dat do paměti (z V/V zařízení)
(−1) způsob posílání tiskových úloh pro tiskárnu
(−1) žádná z výše uvedených možností

Architektura a koncepce OS – OS a HW (archOS_HW)

Aby fungoval OS s preemptivním multitaskingem, musí HW obsahovat:
(+2) přerušovací systém (interrupt system)
(+2) časovač
(−2) řadič SCSI (Small Computer System Interface)
(−2) vícejádrový procesor

(−2) žádná z výše uvedených možností

Při používání DMA:
(+1) se přenosu neúčastní CPU
(+1) je nutné alokovat od systému kanál (DMA)
(−1) se na výpočtu podílí více procesorů
(−1) je nutné použít vícevláknový proces
(−1) žádná z výše uvedených možností

Při používání DMA:
(+2) se urychlí činnost systému, protože se nepoužívá CPU
(+2) je obvyklé používat také přerušovací systém
(−2) se urychlí činnost systému, protože se používá více CPU (nebo HyperThreading)
(−2) je nutné použít vícevláknový proces nebo kooperující procesy
(−2) žádná z výše uvedených možností

Architektura a koncepce OS – Jádro OS (archOS_kernel)

Kolik definuje sysvinit standardně tzv. runlevelů na Linuxu?
(−2) 2
(−2) 4
(−2) 6
(+2) 7
(−2) žádná z výše uvedených možností

Která funkce by měla být povolena pouze v režimu kernel?
(+2) povolení přerušení
(−2) čtení oprávnění k souboru
(+2) změna kořenového adresáře (chroot)
(−2) čtení systémových hodin
(−2) žádná z výše uvedených možností

Která funkce by měla být povolena pouze v režimu kernel?
(+2) zachycení a obsluha interruptu
(−2) zachycení a obsluha zachytitelných signálů
(+2) ovládání V/V zařízení
(−2) tisk prostřednictvím tiskového serveru (subsystému OS)
(−2) žádná z výše uvedených možností

Která funkce by měla být povolena pouze v režimu kernel?
(+2) zákaz přerušení
(−2) čtení času systémových hodin
(+2) nastavení času systémových hodin
(−2) zjištění počtu čekajících procesů
(−2) žádná z výše uvedených možností

Který runlevel sysvinit je v Linuxu standardně určen pro halt (shutdown + power-off)?
(+2) 0
(−2) 1
(−2) 2
(−2) 6
(−2) žádná z výše uvedených možností

Který runlevel sysvinit je v Linuxu standardně určen pro shutdown?
(+2) 0
(-2) 1
(-2) 2
(-2) 6
(-2) žádná z výše uvedených možností

Který runlevel sysvinit je v Linuxu standardně určen pro správu v jednouživatelském režimu?
(−2) 0
(+2) 1
(−2) 2
(−2) 6
(−2) žádná z výše uvedených možností

Který runlevel sysvinit je v Linuxu standardně určen pro reboot?
(−2) 0
(−2) 1
(−2) 2
(+2) 6
(−2) žádná z výše uvedených možností

Systémové volání:
(+1) slouží procesům ke zpřístupnění funkcí OS
(+1) slouží procesům k ovládání V/V zařízení
(−1) slouží OS zejména k preemptivnímu plánování
(−1) slouží HW k předání dat pro OS
(−1) žádná z výše uvedených možností

TRAP:
(-2) je vyvolání přerušení
(+2) je skok z režimu user do režimu kernel
(+2) se používá pro systémová volání
(-2) je přenutí kontextu mezi procesy
(-2) žádná z výše uvedených možností

Architektura a koncepce OS – Typy OS (archOS_typy)

Mezi distribuované systémy patří:
(−1) Windows 2000 Server
(−1) Red Hat Linux do jádra 2.2
(+1) Beowulf cluster
(+1) ParallelKnoppix
(−1) žádná z výše uvedených možností

Mezi RT-systémy patří:
1. (−1) Windows 2008 Server
2. (−1) Linux
3. (+1) QNX
4. (+1) VxWorks
5. (−1) MINIX 3
6. (−1) žádná z výše uvedených možností

Bezpečnost OS

Bezpečnost OS (security)

Mezi nejčastější útoky na systém patří:
(+1) využití chyby ve službách typu buffer overflow
(+1) hádání uživatelských loginů a jejich hesel slovníkovou metodou
(−1) dešifrování zabezpečených vzdálených přihlášení (login sessions)
(−1) využívání tzv. chyby číslo 2F v jádře OS
(−1) žádná z výše uvedených možností

Simulování přihlašovací obrazovky se nazývá:
(+1) login spoofing
(−1) login cracking
(−1) password guessing
(−1) trojan leaving
(−1) žádná z výše uvedených možností

UNIX používá standadně pro uložení hesel funkci crypt() založenou na algoritmu DES. Jak dlouho řádově trvá odvození původního hesla z uloženého záznamu hesla na běžném PC:
(-1) desítky let
(-1) stovky let
(-1) týdny
(+1) nelze
(-1) žádná z výše uvedených možností

UNIX používá standadně pro uložení hesel funkci crypt() založenou na algoritmu DES. Jak dlouho řádově trvá uhodnutí řádně voleného funkčního hesla na běžném PC, pokud máme k dispozici uložený záznam.
(−2) týdny
(−2) roky až stovky let
(+2) desetitisíce až statisíce let
(−2) nelze
(−2) žádná z výše uvedených možností

UNIX používá standadně pro uložení hesel funkci crypt() založenou na algoritmu DES. Jak dlouho řádově trvá uhodnutí řádně voleného funkčního hesla na běžném PC, pokud máme k dispozici uložený záznam.
1. (−2) týdny
2. (−2) roky až desítky let
3. (+2) tisíce až desetitisíce let
4. (−2) nelze
5. (−2) žádná z výše uvedených možností

Procesy

Procesy – Komunikace procesů (proc_kom)

Vyberte správné tvrzení o rourách:
(+1) slouží ke komunikaci procesů
(-1) jsou velmi složité na používání, je nutná znalost architektury jádra OS
(+1) v posixových systémech se s nimi pracuje obdobně jako se soubory
(-1) prakticky se dnes pro předávání dat mezi procesy téměř nepoužívají
(-1) žádná z výše uvedených možností

Vyberte správné tvrzení o socketech:
(+1) slouží ke komunikaci procesů
(−1) jsou velmi složité na používání, je nutná znalost architektury jádra OS
(+1) v posixových systémech se s nimi pracuje obdobně jako se soubory
(−1) prakticky se dnes používají zřídka
(−1) žádná z výše uvedených možností

Procesy – Plánování (proc_plan)

Hlavní cíle plánování procesů na dávkových systémech jsou:
(−1) nízká odezva uživateli
(+1) minimalizace obratu (turnaround time)
(−1) dodržení (časových) termínů
(+1) maximální zátěž (využití) CPU
(−1) žádná z výše uvedených možností

Hlavní cíle plánování procesů na interaktivních systémech jsou:
(−1) maximalizace počtu dokončených procesů
(+1) nízká latence a odezva
(−1) maximální zátěž (využití) CPU
(+1) proporcionalita (přiměřenost) k očekávání uživatelů
(−1) žádná z výše uvedených možností

Hlavní cíle plánování procesů na real-timeových systémech jsou:
(+1) prediktabilita (předvídatelnost)
(−1) minimalizace obratu (turnaround time)
(−1) maximální zátěž (využití) CPU
(+1) dodržení (časových) termínů
(−1) žádná z výše uvedených možností

Hlavní cíle plánování procesů jsou:
(+1) spravedlnost
(+1) rovnováha zatížení subsystémů
(−1) odlehčení zátěže CPU (kvůli přehřívání)
(−1) přidělování dostatku paměti procesům
(−1) žádná z výše uvedených možností

Procesy – Stavy procesů (proc_stavy)

Sedmistavový model procesu nezahrnuje následující stavy:
(−1) běžící, blokovaný, nový
(−1) připravený, běžící, ukončený
(+1) odložený, spustitelný, spící
(+1) vyčerpaný, naplánovaný, odblokovaný
(−1) žádná z výše uvedených možností

Sedmistavový model procesu zahrnuje (mj.) následující stavy:
(+1) běžící, blokovaný, nový
(+1) připravený, běžící, ukončený
(−1) blokovaný odložený, běžící, spící
(−1) nový, naplánovaný, blokovaný
(−1) žádná z výše uvedených možností

Sedmistavový model procesu zahrnuje (mj.) následující stavy:
(−1) běžící, odložený blokovaný, vypršený (timeout)
(−1) připravený, odložený, ukončený
(+1) blokovaný odložený, běžící, nový
(+1) odložený připravený, připravený, blokovaný
(−1) žádná z výše uvedených možností

Sedmistavový model procesu zahrnuje (mj.) následující stavy:
(−1) běžící, odložený připravený, odblokovaný
(−1) nový, odložený, rozvedený
(+1) blokovaný odložený, běžící, ukončený
(+1) odložený blokovaný, blokovaný, připravený
(−1) žádná z výše uvedených možností

Sedmistavový model procesu zahrnuje (mj.) následující stavy:
(+1) běžící, blokovaný, nový
(+1) připravený, běžící, ukončený
(-1) blokovaný odložený, běžící, spící
(-1) nový, naplánovaný, blokovaný
(-1) žádná z výše uvedených možností
Třístavový model procesu zahrnuje následující stav:
(+1) blokovaný
(+1) připravený
(−1) odložený
(−1) nový
(−1) žádná z výše uvedených možností

Třístavový model procesu zahrnuje následující stav:
(+1) blokovaný
(−1) čekající
(+1) běžící
(−1) nový
(−1) žádná z výše uvedených možností

Procesy – Vlákna (proc_threads)

Nevýhodou implementace vláken bez podpory OS je:
(+1) page-fault způsobí zastavení ostatních vláken
(−1) vysoká režie při volání vláknových funkcí
(+1) nutnost převést blokovaná volání na neblokovaná
(−1) vyžaduje se přechod do režimu kernel
(−1) žádná z výše uvedených možností

Vlákna nesdílejí se zbytkem procesu (s ostatními vlákny):
(+1) zásobník
(+1) stav (kontext)

Vlákna sdílejí se zbytkem procesu:
(−1) registry
(−1) zásobník
(−1) stav
(+1) paměť

(−1) žádná z výše uvedených možností

Sdílení prostředků

Sdílení prostředků – Kritická sekce (sdileni_KS)

Kritická sekce je:
(−2) čas provádění alokace prostředku od OS
(+2) část kódu procesu(ů)
(−2) paměťové místo s nepřímým přístupem k proměnným
(−2) řídí přímý přístup do paměti
(−2) žádná z výše uvedených možností

Monitor jako prostředek ošetření vstupu do kritické sekce:
(−2) je nevhodný, protože používá aktivní čekání
(−2) je nevhodný, protože příliš zvyšuje latenci systému
(−2) nelze použít
(−2) se běžně používá v jazyce C, C++ a Delphi
(+2) žádná z výše uvedených možností

Monitor jako prostředek ošetření vstupu do kritické sekce je:
(−1) prostředek operačního systému
(+1) nástroj programovacího jazyka
(−1) prostředek hardware
(−1) softwarová metoda využívající Petersonova algoritmu
(−1) žádná z výše uvedených možností

Nevýhodou řešení kritické sekce pomocí zákazu přerušení je:
(+1) nemožnost použití na SMP-systémech
(+1) zvyšování latence systému
(−1) dlouhá vstupní a výstupní sekce
(−1) nemožnost implementace na architektuře Intel/AMD x86 (IA32)
(−1) žádná z výše uvedených možností

Řešení vstupu do kritické sekce pomocí předávání zpráv jako prostředku OS:
(−1) nelze použít
(−1) nelze použití na všech typech HW
(+1) používá krátkou vstupní a výstupní sekci
(+1) je výhodné pro používání neaktivního čekání
(−1) žádná z výše uvedených možností

Vstup do kritické sekce lze dostatečně ošetřit pomocí:
(+2) prostředků OS, pomocí semaforu
(+2) prostředků OS, pomocí předávání zpráv
(−2) SW metody, pomocí jedné sdílené proměnné booleovského typu
(−2) pouze HW, CPU musí podporovat instrukci test-and-set
(−2) žádná z výše uvedených možností

Výhodou řešení vstupu do kritické sekce pomocí instrukce typu test-and-set je:
(+1) možnost použití na SMP-systémech
(−1) nepotřebnost používání spin-locks
(+1) jednoduchost použití
(−1) neaktivní čekání

(−1) žádná z výše uvedených možností

Výhodou řešení vstupu do kritické sekce pomocí zákazu přerušení je:
(−1) možnost použití na všech systémech
(−1) zlepšení odezvy systému
(+1) jednoduchost použití
(+1) neaktivní čekání
(−1) žádná z výše uvedených možností

Zbytková sekce je:
1. (−2) čas, kdy proces nealokuje žádné prostředky od OS
2. (+2) část kódu procesu(ů)
3. (−2) část datové části paměti procesu s dynamicky alokovanými proměnnými
4. (−2) závislá na přidělení semaforu od OS
5. (−2) žádná z výše uvedených možností

Sdílení prostředků – Semafory (sdileni_sem)

Semafor v OS neobsahuje:
(−1) čítač (čítací proměnnou)
(−1) funkci signal (up)
(−1) funkci wait (down)
(−1) frontu (proměnnou pro seznam procesů)
(+1) žádná z výše uvedených možností

Semafor v OS obsahuje:
1. (+1) čítač (čítací proměnnou)
2. (+1) funkci signal (up)
3. (+1) funkci wait (down)
4. (+1) frontu (proměnnou pro seznam procesů)

Sdílení prostředků – Synchronizace (sdileni_sync)

Synchronizování procesů tak, aby od bariéry běžely oba současně, lze dosáhnout dostatečně pomocí:
(−2) prostředků OS, pomocí jednoho binárního semaforu
(+2) prostředků OS, pomocí předávání zpráv
(−2) SW metody, pomocí jedné sdílené proměnné booleovského typu
(−2) HW metody, pomocí instrukce zakázání přerušení
(−2) žádná z výše uvedených možností

Správa paměti – Pojmy o paměti (mem_teorie)

Položka stránkové tabulky obsahuje:
(−1) číslo stránky
(+1) číslo rámce
(+1) řídicí bity
(−1) velikost stránky
(−1) žádná z výše uvedených možností

Položka segmentové tabulky neobsahuje:
1. (+1) číslo segmentu
4. (+1) offset od bázové adresy

Segmentace:
(+1) usnadňuje sdílení paměti mezi procesy
(-1) není viditelná pro programátora (je transparentní)
(+1) pomáhá implicitně řešit problém ochrany
(-1) používá lineární adresu společnou všem částem programu
(-1) žádná z výše uvedených možností

Stránkování paměti:
(+1) odstraňuje vnější fragmentaci
(−1) odstraňuje vnitřní fragmentaci
(+1) je pro programátora transparentní
(−1) není pro programátora transparentní
(−1) žádná z výše uvedených možností

Thrashing:
(−1) je obecné pojmenování startu OS (boot)
(+1) je neefektivní využití CPU při neustálé výměně paměťových stránek
(+1) může být způsobován odkládáním paměti na disk, když je tato část za okamžik potřebná
(−1) metoda ničení hard disků kvůli bezpečnosti
(−1) žádná z výše uvedených možností

Vnější fragmentace paměti:
(−2) znamená, že paměť procesu je v nesouvislých blocích
(+1) je odstraněna použitím stránkování
(−1) vzniká při přidělení paměti procesu, který její část nevyužije
(−1) je metoda obrany před přetížením řadiče operační paměti
(−1) žádná z výše uvedených možností

Specifické OS

Specifické OS – Systémy reálného času (otherOS_RT)

Mezi typické vlastnosti RTOS nepatří:
(−1) rychlé přepínání kontextu
(+1) nepreemptivní plánování
(−1) multitasking
(+1) plánování zaměřené na maximální využití CPU
(−1) žádná z výše uvedených možností

Mezi typické vlastnosti RTOS patří:
(+1) rychlé přepínání kontextu
(−1) nepreemptivní plánování
(+1) multitasking
(−1) plánování zaměřené na maximální využití CPU
(−1) žádná z výše uvedených možností

Specifické OS – Vestavěné systémy (otherOS_embed)

Podíl trhu mikročipů mimo vestavěné systémy je zhruba:
(+1) < 5 %
(−1) 30 %
(−1) 70 %

(−1) > 90 %
(−1) žádná z výše uvedených možností

Podíl trhu aplikací pro vestavěné systémy je v oblasti telekomunikací a sítí zhruba:
(−1) < 1/6
(+1) 1/5
(−1) 1/3
(−1) > 2/3
(−1) žádná z výše uvedených možností

Podíl trhu mikročipů pro vestavěné systémy je zhruba:
(−1) < 10 %
(−1) 20 %
(−1) 60 %
(+1) > 90 %
(−1) žádná z výše uvedených možností

Architektura a koncepce OS (archOS)
Architektura a koncepce OS – Funkce OS (archOS_fce)
Hlavní funkce OS jsou:
(+1) správa prostředků
(+1) abstrakce a rozšíření počítače
(-1) grafické uživatelské rozhraní
(-1) prioritní řazení procesů
(-1) žádná z výše uvedených možností

Hlavní funkce OS jsou:
(+1) management zdrojů
(+1) virtualizace a rozšíření HW
(-1) GUI
(-1) nepreemptivní plánování procesů
(-1) žádná z výše uvedených možností

Multiprogramingem můžeme označit:
(-1) programování v týmu
(-1) programování aplikací pro audio a video
(+1) způsob práce plánovače OS
(+1) (pseudo)paralelní běh více úloh
(-1) žádná z výše uvedených možností

Multiprogrammingem můžeme označit:
(-1) programování více programátory
(-1) programování multimediálních aplikací
(+1) jeden ze způsobů práce plánovače OS
(+1) (pseudo)současný běh více procesů
(-1) žádná z výše uvedených možností

Pod pojmem spooling rozumíme v oblasti OS také:
(+1) techniku ukládání úloh do fronty pro dávkové systémy
(+1) odkládání dat pro pomalejší V/V zařízení
(-1) algoritmus přidělování paměti vláknům
(-1) sdílení paměti mezi V/V zařízeními
(-1) žádná z výše uvedených možností

Timesharing je:
(+1) způsob multiprogrammingu
(+1) sdílení (dělení) času CPU mezi procesy uživatelů OS
(-1) úspora času při kopírování dat do paměti (z V/V zařízení)
(-1) způsob posílání tiskových úloh pro tiskárnu
(-1) žádná z výše uvedených možností

Architektura a koncepce OS – OS a HW (archOS_HW)
Aby fungoval OS s preemptivním multitaskingem, musí HW obsahovat:
(+2) přerušovací systém (interrupt system)
(+2) časovač
(-2) řadič SCSI (Small Computer System Interface)
(-2) vícejádrový procesor
(-2) žádná z výše uvedených možností

Při používání DMA:
(+1) se přenosu neúčastní CPU
(+1) je nutné alokovat od systému kanál (DMA)
(-1) se na výpočtu podílí více procesorů
(-1) je nutné použít vícevláknový proces
(-1) žádná z výše uvedených možností

Při používání DMA:
(+2) se urychlí činnost systému, protože se nepoužívá CPU
(+2) je obvyklé používat také přerušovací systém
(-2) se urychlí činnost systému, protože se používá více CPU (nebo HyperThreading)
(-2) je nutné použít vícevláknový proces nebo kooperující procesy
(-2) žádná z výše uvedených možností

Architektura a koncepce OS – Jádro OS (archOS_kernel)
Kolik definuje sysvinit standardně tzv. runlevelů na Linuxu?
(-2) 2
(-2) 4
(-2) 6
(+2) 7
(-2) žádná z výše uvedených možností

Která funkce by měla být povolena pouze v režimu kernel?
(+2) povolení přerušení
(-2) čtení oprávnění k souboru
(+2) změna kořenového adresáře (chroot)
(-2) čtení systémových hodin
(-2) žádná z výše uvedených možností

Která funkce by měla být povolena pouze v režimu kernel?
(+2) zachycení a obsluha interruptu
(-2) zachycení a obsluha zachytitelných signálů
(+2) ovládání V/V zařízení
(-2) tisk prostřednictvím tiskového serveru (subsystému OS)

(-2) žádná z výše uvedených možností

Která funkce by měla být povolena pouze v režimu kernel?
(+2) zákaz přerušení
(-2) čtení času systémových hodin
(+2) nastavení času systémových hodin
(-2) zjištění počtu čekajících procesů
(-2) žádná z výše uvedených možností

Který runlevel sysvinit je v Linuxu standardně určen pro halt (shutdown + power-off)?
(+2) 0
(-2) 1
(-2) 2
(-2) 6
(-2) žádná z výše uvedených možností

Který runlevel sysvinit je v Linuxu standardně určen pro shutdown?
(+2) 0
(-2) 1
(-2) 2
(-2) 6
(-2) žádná z výše uvedených možností

Který runlevel sysvinit je v Linuxu standardně určen pro správu v jednouživatelském režimu?
(-2) 0
(+2) 1
(-2) 2
(-2) 6
(-2) žádná z výše uvedených možností

Který runlevel sysvinit je v Linuxu standardně určen pro reboot?
(-2) 0
(-2) 1
(-2) 2
(+2) 6
(-2) žádná z výše uvedených možností

Systémové volání:
(+1) slouží procesům ke zpřístupnění funkcí OS
(+1) slouží procesům k ovládání V/V zařízení
(-1) slouží OS zejména k preemptivnímu plánování
(-1) slouží HW k předání dat pro OS

(-1) žádná z výše uvedených možností

TRAP:
(-2) je vyvolání přerušení
(+2) je skok z režimu user do režimu kernel
(+2) se používá pro systémová volání
(-2) je přenutí kontextu mezi procesy
(-2) žádná z výše uvedených možností

Architektura a koncepce OS – Typy OS (archOS_typy)
Mezi distribuované systémy patří:
(-1) Windows 2000 Server
(-1) Red Hat Linux do jádra 2.2
(+1) Beowulf cluster
(+1) ParallelKnoppix
(-1) žádná z výše uvedených možností

Mezi RT-systémy patří:
1. (-1) Windows 2008 Server
2. (-1) Linux
3. (+1) QNX
4. (+1) VxWorks
5. (-1) MINIX 3
6. (-1) žádná z výše uvedených možností

Bezpečnost OS
Bezpečnost OS (security)
Mezi nejčastější útoky na systém patří:
(+1) využití chyby ve službách typu buffer overflow
(+1) hádání uživatelských loginů a jejich hesel slovníkovou metodou
(-1) dešifrování zabezpečených vzdálených přihlášení (login sessions)
(-1) využívání tzv. chyby číslo 2F v jádře OS
(-1) žádná z výše uvedených možností

Simulování přihlašovací obrazovky se nazývá:
(+1) login spoofing
(-1) login cracking
(-1) password guessing
(-1) trojan leaving
(-1) žádná z výše uvedených možností

UNIX používá standadně pro uložení hesel funkci crypt() založenou na algoritmu DES. Jak dlouho řádově trvá odvození původního hesla z uloženého záznamu hesla na běžném PC:

(-1) desítky let
(-1) stovky let
(-1) týdny
(+1) nelze
(-1) žádná z výše uvedených možností

UNIX používá standadně pro uložení hesel funkci crypt() založenou na algoritmu DES. Jak dlouho řádově trvá uhodnutí řádně voleného funkčního hesla na běžném PC, pokud máme k dispozici uložený záznam.
(-2) týdny
(-2) roky až stovky let
(+2) desetitisíce až statisíce let
(-2) nelze
(-2) žádná z výše uvedených možností

UNIX používá standadně pro uložení hesel funkci crypt() založenou na algoritmu DES. Jak dlouho řádově trvá uhodnutí řádně voleného funkčního hesla na běžném PC, pokud máme k dispozici uložený záznam.

1. (-2) týdny
2. (-2) roky až desítky let
3. (+2) tisíce až desetitisíce let
4. (-2) nelze
5. (-2) žádná z výše uvedených možností

Procesy
Procesy – Komunikace procesů (proc_kom)
Vyberte správné tvrzení o rourách:
(+1) slouží ke komunikaci procesů
(-1) jsou velmi složité na používání, je nutná znalost architektury jádra OS
(+1) v posixových systémech se s nimi pracuje obdobně jako se soubory
(-1) prakticky se dnes pro předávání dat mezi procesy téměř nepoužívají
(-1) žádná z výše uvedených možností

Vyberte správné tvrzení o socketech:
(+1) slouží ke komunikaci procesů
(-1) jsou velmi složité na používání, je nutná znalost architektury jádra OS
(+1) v posixových systémech se s nimi pracuje obdobně jako se soubory
(-1) prakticky se dnes používají zřídka
(-1) žádná z výše uvedených možností

Procesy – Plánování (proc_plan)
Hlavní cíle plánování procesů na dávkových systémech jsou:

(-1) nízká odezva uživateli
(+1) minimalizace obratu (turnaround time)
(-1) dodržení (časových) termínů
(+1) maximální zátěž (využití) CPU
(-1) žádná z výše uvedených možností

Hlavní cíle plánování procesů na interaktivních systémech jsou:
(-1) maximalizace počtu dokončených procesů
(+1) nízká latence a odezva
(-1) maximální zátěž (využití) CPU
(+1) proporcionalita (přiměřenost) k očekávání uživatelů
(-1) žádná z výše uvedených možností

Hlavní cíle plánování procesů na real-timeových systémech jsou:
(+1) prediktabilita (předvídatelnost)
(-1) minimalizace obratu (turnaround time)
(-1) maximální zátěž (využití) CPU
(+1) dodržení (časových) termínů
(-1) žádná z výše uvedených možností

Hlavní cíle plánování procesů jsou:
(+1) spravedlnost
(+1) rovnováha zatížení subsystémů
(-1) odlehčení zátěže CPU (kvůli přehřívání)
(-1) přidělování dostatku paměti procesům
(-1) žádná z výše uvedených možností

Procesy – Stavy procesů (proc_stavy)
Sedmistavový model procesu nezahrnuje následující stavy:
(-1) běžící, blokovaný, nový
(-1) připravený, běžící, ukončený
(+1) odložený, spustitelný, spící
(+1) vyčerpaný, naplánovaný, odblokovaný
(-1) žádná z výše uvedených možností

Sedmistavový model procesu zahrnuje (mj.) následující stavy:
(+1) běžící, blokovaný, nový
(+1) připravený, běžící, ukončený
(-1) blokovaný odložený, běžící, spící
(-1) nový, naplánovaný, blokovaný
(-1) žádná z výše uvedených možností

Sedmistavový model procesu zahrnuje (mj.) následující stavy:

(-1) běžící, odložený blokovaný, vypršený (timeout)
(-1) připravený, odložený, ukončený
(+1) blokovaný odložený, běžící, nový
(+1) odložený připravený, připravený, blokovaný
(-1) žádná z výše uvedených možností

Sedmistavový model procesu zahrnuje (mj.) následující stavy:
(-1) běžící, odložený připravený, odblokovaný
(-1) nový, odložený, rozvedený
(+1) blokovaný odložený, běžící, ukončený
(+1) odložený blokovaný, blokovaný, připravený
(-1) žádná z výše uvedených možností

Sedmistavový model procesu zahrnuje (mj.) následující stavy:
(+1) běžící, blokovaný, nový
(+1) připravený, běžící, ukončený
(-1) blokovaný odložený, běžící, spící
(-1) nový, naplánovaný, blokovaný
(-1) žádná z výše uvedených možností
Třístavový model procesu zahrnuje následující stav:
(+1) blokovaný
(+1) připravený
(-1) odložený
(-1) nový
(-1) žádná z výše uvedených možností

Třístavový model procesu zahrnuje následující stav:
(+1) blokovaný
(-1) čekající
(+1) běžící
(-1) nový
(-1) žádná z výše uvedených možností

Procesy – Vlákna (proc_threads)
Nevýhodou implementace vláken bez podpory OS je:
(+1) page-fault způsobí zastavení ostatních vláken
(-1) vysoká režie při volání vláknových funkcí
(+1) nutnost převést blokovaná volání na neblokovaná
(-1) vyžaduje se přechod do režimu kernel
(-1) žádná z výše uvedených možností

Vlákna nesdílejí se zbytkem procesu (s ostatními vlákny):
(+1) zásobník

(+1) stav (kontext)

Vlákna sdílejí se zbytkem procesu:
(-1) registry
(-1) zásobník
(-1) stav
(+1) paměť
(-1) žádná z výše uvedených možností

Sdílení prostředků
Sdílení prostředků – Kritická sekce (sdileni_KS)
Kritická sekce je:
(-2) čas provádění alokace prostředku od OS
(+2) část kódu procesu(ů)
(-2) paměťové místo s nepřímým přístupem k proměnným
(-2) řídí přímý přístup do paměti
(-2) žádná z výše uvedených možností

Monitor jako prostředek ošetření vstupu do kritické sekce:
(-2) je nevhodný, protože používá aktivní čekání
(-2) je nevhodný, protože příliš zvyšuje latenci systému
(-2) nelze použít
(-2) se běžně používá v jazyce C, C++ a Delphi
(+2) žádná z výše uvedených možností

Monitor jako prostředek ošetření vstupu do kritické sekce je:
(-1) prostředek operačního systému
(+1) nástroj programovacího jazyka
(-1) prostředek hardware
(-1) softwarová metoda využívající Petersonova algoritmu
(-1) žádná z výše uvedených možností

Nevýhodou řešení kritické sekce pomocí zákazu přerušení je:
(+1) nemožnost použití na SMP-systémech
(+1) zvyšování latence systému
(-1) dlouhá vstupní a výstupní sekce
(-1) nemožnost implementace na architektuře Intel/AMD x86 (IA32)
(-1) žádná z výše uvedených možností

Řešení vstupu do kritické sekce pomocí předávání zpráv jako prostředku OS:
(-1) nelze použít
(-1) nelze použití na všech typech HW
(+1) používá krátkou vstupní a výstupní sekci

(+1) je výhodné pro používání neaktivního čekání
(-1) žádná z výše uvedených možností

Vstup do kritické sekce lze dostatečně ošetřit pomocí:
(+2) prostředků OS, pomocí semaforu
(+2) prostředků OS, pomocí předávání zpráv
(-2) SW metody, pomocí jedné sdílené proměnné booleovského typu
(-2) pouze HW, CPU musí podporovat instrukci test-and-set
(-2) žádná z výše uvedených možností

Výhodou řešení vstupu do kritické sekce pomocí instrukce typu test-and-set je:
(+1) možnost použití na SMP-systémech
(-1) nepotřebnost používání spin-locks
(+1) jednoduchost použití
(-1) neaktivní čekání
(-1) žádná z výše uvedených možností

Výhodou řešení vstupu do kritické sekce pomocí zákazu přerušení je:
(-1) možnost použití na všech systémech
(-1) zlepšení odezvy systému
(+1) jednoduchost použití
(+1) neaktivní čekání
(-1) žádná z výše uvedených možností

Zbytková sekce je:
1. (-2) čas, kdy proces nealokuje žádné prostředky od OS
2. (+2) část kódu procesu(ů)
3. (-2) část datové části paměti procesu s dynamicky alokovanými proměnnými
4. (-2) závislá na přidělení semaforu od OS
5. (-2) žádná z výše uvedených možností

Sdílení prostředků – Semafory (sdileni_sem)
Semafor v OS neobsahuje:
(-1) čítač (čítací proměnnou)
(-1) funkci signal (up)
(-1) funkci wait (down)
(-1) frontu (proměnnou pro seznam procesů)
(+1) žádná z výše uvedených možností

Semafor v OS obsahuje:
1. (+1) čítač (čítací proměnnou)
2. (+1) funkci signal (up)
3. (+1) funkci wait (down)

4. (+1) frontu (proměnnou pro seznam procesů)

Sdílení prostředků – Synchronizace (sdileni_sync)
Synchronizování procesů tak, aby od bariéry běžely oba současně, lze dosáhnout dostatečně pomocí:

(-2) prostředků OS, pomocí jednoho binárního semaforu
(+2) prostředků OS, pomocí předávání zpráv
(-2) SW metody, pomocí jedné sdílené proměnné booleovského typu
(-2) HW metody, pomocí instrukce zakázání přerušení
(-2) žádná z výše uvedených možností

Správa paměti – Pojmy o paměti (mem_teorie)
Položka stránkové tabulky obsahuje:
(-1) číslo stránky
(+1) číslo rámce
(+1) řídicí bity
(-1) velikost stránky
(-1) žádná z výše uvedených možností

Položka segmentové tabulky neobsahuje:
1. (+1) číslo segmentu
4. (+1) offset od bázové adresy

Segmentace:
(+1) usnadňuje sdílení paměti mezi procesy
(-1) není viditelná pro programátora (je transparentní)
(+1) pomáhá implicitně řešit problém ochrany
(-1) používá lineární adresu společnou všem částem programu
(-1) žádná z výše uvedených možností

Stránkování paměti:
(+1) odstraňuje vnější fragmentaci
(-1) odstraňuje vnitřní fragmentaci
(+1) je pro programátora transparentní
(-1) není pro programátora transparentní
(-1) žádná z výše uvedených možností

Thrashing:
(-1) je obecné pojmenování startu OS (boot)
(+1) je neefektivní využití CPU při neustálé výměně paměťových stránek
(+1) může být způsobován odkládáním paměti na disk, když je tato část za okamžik potřebná

(-1) metoda ničení hard disků kvůli bezpečnosti
(-1) žádná z výše uvedených možností
Vnější fragmentace paměti:
(-2) znamená, že paměť procesu je v nesouvislých blocích
(+1) je odstraněna použitím stránkování
(-1) vzniká při přidělení paměti procesu, který její část nevyužije
(-1) je metoda obrany před přetížením řadiče operační paměti
(-1) žádná z výše uvedených možností

Specifické OS
Specifické OS – Systémy reálného času (otherOS_RT)
Mezi typické vlastnosti RTOS nepatří:
(-1) rychlé přepínání kontextu
(+1) nepreemptivní plánování
(-1) multitasking
(+1) plánování zaměřené na maximální využití CPU
(-1) žádná z výše uvedených možností

Mezi typické vlastnosti RTOS patří:
(+1) rychlé přepínání kontextu
(-1) nepreemptivní plánování
(+1) multitasking
(-1) plánování zaměřené na maximální využití CPU
(-1) žádná z výše uvedených možností

Specifické OS – Vestavěné systémy (otherOS_embed)
Podíl trhu mikročipů mimo vestavěné systémy je zhruba:
(+1) < 5 %
(-1) 30 %
(-1) 70 %
(-1) > 90 %
(-1) žádná z výše uvedených možností

Podíl trhu aplikací pro vestavěné systémy je v oblasti telekomunikací a sítí zhruba:
(-1) < 1/6
(+1) 1/5
(-1) 1/3
(-1) > 2/3
(-1) žádná z výše uvedených možností

Podíl trhu mikročipů pro vestavěné systémy je zhruba:
(-1) < 10 %

(-1) 20 %
(-1) 60 %
(+1) > 90 %
(-1) žádná z výše uvedených možností