Písemná zkouška (skupina A) z předmětu Operační systémy (IOS)

17.5.2013 8:00-10:00 (60 bodů)

Jméno a přijmení: …..... Přihlašovací jméno: ….....

Hodnocení	1	2	3	4	5	6	∑

Obecné pokyny:

Není-li explicitně řečeno jinak, předpokládáme operační systém UNIX. U vašich odpovědí se snažte o maximální přesnost a výstižnost. Pište a kreslete čitelně a jednoznačně. V části ODPOVĚZTE STRUČNĚ dodržte délky limity odpovědí: výrazné nedodržení může být chápáno jako nesprávná odpověď! Větou se přitom rozumí **souvislá posloupnost slov, začínající velkým písmenem a končící tečkou**; v žádném případě nejde o změť slov či symbolů umístěných volně na určité ploše!!! Podobně, je-li vyžadován **pseudokód nebo kód**, bude odpověď ve formě volného textu hodnocena zásadně nula body bez ohledu na její délku a obsah!!! Na závěr vložte podepsané papíry (A4) na sebe, nahoru položte zadání a vše přeložte na polovinu (tj na formát A5) tak, aby byl tento text (jméno a přijmení) nahoře. Podepsané zadání musíte odevzdat, jinak nebude řešení uznáno.

1. Odpovězte stručně

1. (6 bodů) Jaký je maximální počet čtení bloku z disku při provedení operací `h = open(„/symlink“, O_RDONLY); read(h, buf, 10); read(h, buf, 10);`
Předpokládejte přitom, že adresáře jsou kratší, než 1 blok, `symlink` je symbolický odkaz na soubor `/file`, file je klasický soubor delší, než 20B, žádný blok není na počátku ve vyrovnávací paměti a používají se všechny běžně používané vrovnávací paměti, nedochází k interferenci s dalšími procesy, přepnutí kontextu, příchod signálů apod.
(limity: 1 číslo [bez zdůvodnění za 0b] a cca 6 mírně rozvitých vět)

2. (6 bodů) Charakterizujte základní myšlenku a princip činnosti plánovače CFS (Completely Fair Scheduler) implementovaného v Linuxu 2.6.
(limit cca 6 rozvitých vět)

3. (9 bodů) Uvažujme strukturu `struct sMonitor` se členy `semaphore mutex`, `semaphore condition` a `int waiting`, kterou hodláme použít pro simulaci monitoru v jazyce C, který monitory přímo nepodporuje. Předpokládáme přitom, že daný monitor bude mít napevnojednu podmínku pro čekání „uvnitř“ monitoru, pro jejíž implementaci bude použit semafor `condition` a čítač `waiting`. Simulace bude probíhat tak, že na začátku každé operace, která má být monitorem chráněna, bude volána funkce `void enter(struct sMonitor *monitor)` a na konci každé takové operace bude voláno `void leave(struct sMonitor *monitor)`. S čekací podmínkou bude možno pracovat pomocí funkcí `void wait(struct sMonitor *monitor)` a `void notify(struct sMonitor *monitor)`, u kterých předpokládáme, že jsou vždy volány mezi voláním `enter` a `leave` na patřičném monitoru. U funkce `notify` předpokládáme, že pokud nikdo na notifikaci nečeká, jedná se o prázdnou operaci. Poradí notifikovaných procesů není podstatné. Implementujte funkce `enter, leave, wait` a `notify`.
(limity: nejdelší z funkcí cca 4 řádky kódu)

4. (8 bodů) Předpokládejte existenci struktury `struct inv_tab_item` popisující v jazyce C položku invertované tabulky stránek. Dále nechť číselný typ `page_num_t` popisuje čísla stránek a rámců a číselný typ `pid_t` čísla procesů. Implementujte v jazyce C funkci pro převod čísla stránky

na číslo rámce s prototypem
`page_num_t page_to_frame(inv_tab_item *inv_tab, page_num_t max_frame_num, pid_t pid, page_num_t page_num);`
Předpokládejte přito, že parametr `inv_tab` ukazuje na první položku invertované tabulky stránek, `max_frame_num` je nejvyšší možné číslo rámce v systému, `pid` je číslo procesu, který převod provádí a `page_num` je číslo převáděné stránky. Funkce v případě, kdy je možno dané číslo stránky převést na odpovídající číslo rámce, vrací příslušné číslo rámce. Pokud převod není možno provést, funkce vrátí hodnotu `PAGE_FAULT`. Pro potřeby implementace fce `page_to_frame` si doplňte potřebné členy struktury `inv_tab_item`, Popište, se kterými v implementaci počítáte. (limity: patřičný počet členů struktury inv_tab_item, 4 řádky kódu pro tělo funkce bez deklarací)

---

2. ROZVEĎTE

5. (15 bodů) Definujte pojem uváznutí (deadlock) při práci se sdílenými zdroji a uveďte 4 nutné (tzv. Coffmanovy podmínky) jeho vzniku. Uveďte základní princip prevence uváznutí, popište co nejdetailněji různé přístupy, které se k tomuto účelu používají. Uveďte základní princip vyhýbání se uváznutí a zvláštnípozornost pak věnujte pečlivému popisu grafu alokaci zdrojů a jeho použití pro vyhýbání se a uváznutí. Jaký další klasický mechanismus řešení problému uváznutí se používá (neuvažujeme-li využití verifikace k ověření, že uváznutí nemůže nastat)? Stručně charakterizujte.

6. (16 bodů) Co musí udělat operační systém při výpadku stránky? (Popište co nejpřesněji všechny možné varianty.) Jaká je základní myšlenka algoritmu FIFO, jeho výhody a nevýhody a v jaké podobě je možné ho v praxi používat? Jaká je základní myšlenka algoritmu LRU pro výběr tzv. *victim page*, jeho základní výhoda a nevýhody a základní princip dvou variant, ve kterých se běžně užívá v praxi.