Test z předmětu Operační systémy 2018/2019

U každé otázky jsou v hranatých závorkách uvedeny body za úplné zodpovězení. Celkem nejvýše 20 bodů.

1. **Synchronizace [5]**

Implementujte spin lock. Z operací, které mají zaručené chování při paralelním vykonání více procesory nad stejnými daty, máte k dispozici (pouze) tyto:

- `int atomic_read (int *p)` pro atomické čtení proměnné z paměti,
- `void atomic_write (int *p, int v)` pro atomický zápis proměnné do paměti,
- `int atomic_read_add (int *p, int v)` pro atomické přičtení hodnoty, které vrací původní hodnotu.

Předpokládejte, že váš spin lock budou používat tři vlákna, která budou všechna současně (každé na svém procesoru) vykonávat následující cyklus nad sdíleným zámkem `l` a sdílenou proměnnou `i` s atributem `volatile` (tedy proměnná bude mezi jednotlivými kroky programu uložená v paměti a ne jen v registru):

```
while (true) {
    lock (l);
    i ++;
    unlock (l);
}
```

Zkuste hrubě odhadnout, jak rychle se bude zvětšovat hodnota proměnné `i` oproti situaci, kdy by pouze jedno vlákno vykonávalo cyklus `while (true) i++;`. U odhadu se očekává hlavně zdůvodnění (co a proč bude kolikrát rychlejší nebo pomalejší), konkrétní hodnoty mají pouze rámcový význam.

2. **Stránkování [5]**

Uvažujte systém s délkou virtuální adresy 32 bitů a délkou fyzické adresy 32 bitů, s překladem pomocí TLB plněné operačním systémem. Zvolte detailní strukturu položky TLB a napište (jako pseudokód nebo vzorec) způsob překladu virtuální adresy na fyzickou, včetně ošetření výjimečných situací.

Spočítejte, kolik výpadků ve vámi zvolené TLB může v nejlepším a v nejhorším případě způsobit program, který sekvenčně projde zřetězený seznam o 10 položkách, kde každá položka se skládá z jedné programem čtené hodnoty typu `int` a jednoho ukazatele na další položku, poslední položka má tento ukazatel nastaven na `NULL`. Uvažujte pouze přístup k datům, nikoliv k instrukcím programu.

3. **Plánování [2]**

Definujte význam základních stavů procesu (ready, running, waiting, zombie) a nakreslete diagram vyznačující možné přechody mezi těmito stavy. U každého přechodu napište, za jakých situací k němu může dojít.

4. **Správa paměti [2]**

Uvažujte heap alokátor bez možnosti přesunu alokovaných bloků, u kterého můžete zanedbat režii na alokaci bloku (jako kdyby alokované bloky neměly hlavičky a nebyly zarovnávané). Máte k dispozici 1000000 bajtů heapu. Navrhněte postup (načrtněte pseudokód), kterým na tomto heapu obsadíte co nejméně paměti, ale přitom již nepůjde alokovat žádný blok s velikostí 100 bajtů nebo více. Nemusíte řešit, kam uložit případné pomocné datové struktury.

Otázky i na druhé straně listu !

5. **Správa zařízení [2]**

Uvažujte ovladač disku, který nabízí rozhraní s metodami pro čtení a zápis jednoho sektoru. Ovladač může volat více procesů, tedy v jednom okamžiku může čekat na obsluhu několik požadavků na čtení a zápis. Navrhněte strategii obsluhy těchto požadavků, pokud víte, že disk je mechanický a jednotlivé přístupy tedy mají latenci složenou s času na přesun diskové hlavy na určenou stopu a z času na přečtení daného sektoru z určené stopy.

6. **Systémy souborů [2]**

Aplikace potřebuje ukládat na disk zhruba milion souborů s délkou pouze 16 bajtů, jména souborů jsou devítimístná čísla. Zvolte si systém souborů a odhadněte diskový prostor obsazený tímto množstvím souborů (všechen prostor, který po uložení souborů již nebude k dispozici pro jiné využití).

7. **Vlákna [2]**

Navrhněte rozhraní (včetně popisu významu parametrů), kterým může jedno vlákno procesu spustit a počkat na ukončení jiného vlákna.