Systemy operacyjne 2016

Lista zadań nr 5

Na zajęcia 3 listopada 2016

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Stallings (wydanie ósme): 5.1 - 5.5, 6.1, 6.2
- Tanenbaum (wydanie czwarte): 2.3

UWAGA! W trakcie prezentacji rozwiązań należy zdefiniować i wyjaśnić pojęcia, które zostały oznaczone **wytłuszczoną** czcionką.

Zadania wymagające użycia rzutnika, oznaczenie **(P)**, należy starannie przygotować do prezentacji przed zajęciami. Można nie otrzymać punktów za zadanie w przypadku zbędnego przeciągania czasu odpowiedzi ze względu na problemy techniczne.

Zadanie 1. Podaj w pseudokodzie przykład szkodliwej **rywalizacji** procesów o dostęp do współdzielonych danych. Programy, w których występuje **sytuacja wyścigu** (ang. *race condition*), są szczególnie uciążliwe w odpluskwianiu – dlaczego? Zdefiniuj i odnieś się do pojęcia Heisenbug[1].

Zadanie 2. Wymień cztery warunki konieczne do zaistnienia **zakleszczenia** (ang. *deadlock*). Czym różni się zakleszczenie od **niejawnego zakleszczenia** (ang. *livelock*) i **głodzenia** (ang. *starvation*)? W jaki sposób programista może przeciwdziałać zakleszczeniom (ang. *deadlock prevention*)?

Zadanie 3 (P). Podaj i uzasadnij założenia jakie musi spełniać rozwiązanie problemu **sekcji krytycznej**. Porównaj sprzętowe mechanizmy umożliwiające implementację sekcji krytycznej: wyłączenie przerwań, **instrukcje atomowe**, **pamięć transakcyjna**. Podaj w pseudokodzie semantykę instrukcji atomowej **compare-and-swap** i z jej użyciem zaimplementuj **blokadę wirującą** (ang. *spin-lock*).

Zadanie 4. W tym ćwiczeniu zapoznamy się wstępnie z różnymi środkami **komunikacji międzyprocesowej** (ang. *Inter-Process Communication*). Przykłady ich zastosowań w projektowaniu oprogramowania przedstawiono w The Art of Unix Programming: Taxonomy of Unix IPC Methods[2]

(a) Jaką funkcję realizują **semafory binarne** i **semafory zliczające**? Czym charakteryzują się **semafory silne** (ang. *strong semaphore*)? Podaj semantykę operacji **signal** i **wait**. Czym różni się **mutex** od semafora binarnego?
(b) Jaką funkcję realizują **potoki** (ang. *pipe*) i **gniazda** (ang. *socket*). Czym różnią się **gniazda datagramowe** (ang. *datagram socket*) od **strumieniowych** (ang. *stream socket*)?

Zadanie 5 (P). Proces, który próbuje wejść pod opuszczony semafor musi zaczekać do momentu, w którym inny proces podniesie semafor. Może w tym celu użyć **aktywnego czekania** (ang. *busy-waiting*), albo poprosić jądro o **uśpienie** procesu na czas oczekiwania.

Porównaj te rozwiązania i opisz rozwiązanie pośrednie, czyli **semafory adaptacyjne**. Jaką rolę pełni mechanizm **futex(2)** w systemie `Linux`? Z użyciem operacji `FUTEX_WAIT` i `FUTEX_WAKE` podaj w pseudokodzie szkic implementacji funkcji `wait` i `signal` dla semaforów binarnych.

[1] `http://www.catb.org/~esr/jargon/html/H/heisenbug.html`
[2] `http://www.catb.org/esr/writings/taoup/html/ch07s02.html`

Zadanie 6 (P). Zaprezentuj przykład zastosowania narzędzia do programowania współbieżnego zwanego **monitorem**. Wytłumacz różnice między **monitorami Hoare'a**, a **monitorami Mesa**.

Systemy operacyjne udostępniają mechanizm **zmiennych warunkowych** do implementacji monitorów. Jakie dane przechowuje zmienna warunkowa? Podaj sygnaturę i semantykę operacji **wait**, **signal** i **broadcast** dla zmiennych warunkowych. Używając zmiennych warunkowych podaj w pseudokodzie szkic implementacji semafora zliczającego.

Zadanie 7. Mechanizm **przekazywania komunikatów** wymaga implementacji co najmniej dwóch funkcji: `send(dest,msg)` i `recv(src,msg)`. Jak wygląda struktura komunikatu? Jakie problemy stwarza synchronizacja z użyciem przekazywania komunikatów porównując z semaforami i pamięcią dzieloną? Podaj warianty semantyki operacji `send` i `recv` dla **skrzynek pocztowych**. Jak adresować **nadawcę** lub **odbiorcę**? Czym charakteryzują się **punkty schadzek** (ang. *rendezvous*)?

Zadanie 8. Jedną z technik wymuszania spójności współdzielonych struktur danych bez stosowania blokad jest **RCU** (ang. *read-copy-update*). Działa ona przy założeniu, że strukturę danych może przeglądać wiele wątków czytających i aktualizować co najwyżej jeden piszący. Jakie dodatkowe założenia należy przyjąć w stosunku do wątków czytających? W jaki sposób wykorzystywany jest fakt oddzielenia **fazy usuwania** elementów od **fazy odzyskiwania** pamięci? Biorąc pod uwagę rozpatrzone ograniczenia podaj przykłady w jakich stosowanie tej techniki jest uzasadnione.