Systemy operacyjne 2016

Lista zadań nr 4

Na zajęcia 27 października 2016

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Stallings (wydanie siódme): 3.4, 4.1 – 4.3, 9.1
- Tanenbaum (wydanie czwarte): 2.1, 2.2, 2.4.1, 10.3

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

Zadania wymagające użycia rzutnika, oznaczenie **(P)**, należy starannie przygotować w domu – najlepiej w postaci pliku tekstowego z listą poleceń do wykonania i komentarzami. Każde zadanie należy mieć właściwie przygotowane do prezentacji przed zajęciami. Można nie otrzymać punktów za zadanie w przypadku zbędnego przeciągania czasu odpowiedzi ze względu na problemy techniczne.

Zadanie 1 (P). Uruchom aplikację `firefox` i przy pomocy programu `lsof` wyświetl zasoby należące do procesu przeglądarki. Podaj znaczenie poszczególnych kolumn wykazu i zidentyfikuj, które z wymienionych zasobów są **zwykłymi plikami**, **katalogami**, **urządzeniami**, **gniazdami** (sieciowymi lub domeny uniksowej), **potokami**. Przechwyć wyjście z programu `lsof` przed i po otwarciu wybranej strony w nowej zakładce, po czym wyświetl różnice poleceniem `diff -u`.

Zadanie 2. Dlaczego **planowanie zadań** jest z reguły podzielone na dwie części? Z jaką częstotliwością uruchamia się **planista krótko-** i **długoterminowy**? Zauważ, że algorytm planowania nie odpowiada w sposób bezpośredni za wymianę (ang. *swapping*) procesów na dysk! Które zdarzenia i zużycie których zasobów należałoby **monitorować** (ang. *accounting*[1]), by wspomóc decyzje podejmowane przez planistę długoterminowego?

Zadanie 3. Czym różni się **przetwarzanie równoległe** (ang. *parallel*) od **przetwarzania współbieżnego** (ang. *concurrent*)? Czym charakteryzują się **funkcje wielobieżne** (ang. *reentrant*)?

Zadanie 4. Opisz przebieg **przełączania procesów** w systemie `Linux` lub `FreeBSD`. Czemu w trakcie **przełączania przestrzeni adresowych** należy opróżnić `TLB` (ang. *Translation Lookaside Buffer*)? Czym różni się **przełączanie kontekstu** od **przełączania trybu pracy**? Gdzie jądro przechowuje kontekst przy przejściu z przestrzeni użytkownika do przestrzeni jądra? Czemu każdy wątek posiada odrębny stos w przestrzeni jądra?

Zadanie 5. Opisz różnice między **wątkami przestrzeni jądra** (ang. *kernel-level threads*), a **wątkami przestrzeni użytkownika** (ang. *user-level threads*) – rozważ zalety i wady obu podejść. Jak biblioteka `ULT` kompensuje brak wsparcia jądra dzięki **obwolutowaniu** (ang. *jacketing*)? Opisz model hybrydowy bazujący na **aktywacjach planisty** i pokaż, że może on łączyć zalety `KLT` i `ULT`.

[1] Określenia „księgowanie" będziemy używać w kontekście systemów plików.

Zadanie 6. Najpowszechniej implementowane wątki przestrzeni jądra implikują wiele interesujących niejasności. Czy proces utworzony przy pomocy `fork()` dziedziczy wątki? Czy wątki **współdzielą** globalną zmienną **errno(3)**? Do czego służy **przestrzeń lokalna wątku** (ang. *thread local storage*)? Użytkownik przerywa program z klawiatury – który wątek obsłuży sygnał `SIGINT`? Wątek wykonuje niewłaściwe odwołanie do pamięci – kto odbierze sygnał `SIGSEGV`? Wątki w danym procesie współdzielą stertę – co jeśli wszystkie na raz próbują pobrać blok pamięci za pomocą funkcji `malloc()`?

Zadanie 7. Wątki nie są panaceum na problemy z wydajnością oprogramowania na **maszynach wieloprocesorowych ze współdzieloną pamięcią** (ang. *Shared Memory Processing*). Wymień warunki jakie musi spełniać architektura programu by stosowanie wątków było uzasadnione (§4.3)? Co ogranicza wydajność architektury bazującej na wątkach?

Zadanie 8. Jakie niebezpieczeństwa niesie ze sobą stosowanie wątków? Zapoznaj się z:

- The Art of Unix Programming: Threads – Threat or Menace?[2]
- Why Threads Are A Bad Idea (for most purposes)[3]

... i wyjaśnij czemu w ogóle należałoby unikać stosowania wątków?

Zadanie 9 (bonus). Na przykładzie systemu Linux opisz zgrubnie proces uruchomienia programu z dysku, tj. od wprowadzenia polecenia w powłoce `bash` po wejście do funkcji `main()`. Które z poszczególnych etapów przebiegają po stronie jądra, **dynamicznego konsolidatora**, a za które odpowiada kod uruchamianego programu? Upewnij się, że nie pomijasz żadnego ważnego etapu, tj. tworzenia przestrzeni adresowej, ładowania bibliotek dynamicznych, wywołania **procedury startowej** `crt0`.

[2] `http://www.catb.org/esr/writings/taoup/html/ch07s03.html\#id2923889`
[3] `https://web.stanford.edu/~ouster/cgi-bin/papers/threads.pdf`