Struktura jąder systemów operacyjnych

Lista zadań nr 5

Na zajęcia 22 i 29 kwietnia 2020

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

Zadanie 1. Na podstawie [3, §7.4.1] zreferuj problem **zagubionej pobudki** (ang. *lost wakeup*) charakterystyczny dla implementacji procedury «sleep» w starych systemach UNIX. Czy procedura `mtx_sleep(9)` dostępna w jądrze FREEBSD rozwiązuje ten problem?

Zadanie 2. Na podstawie [3, §7.5.5] zreferuj problem powstawania **konwojów** (ang. *convoy*) na semaforach. Załóżmy, że w wyniku rywalizacji o dostęp do sekcji krytycznej chronionej przez semafor, tworzy się konwój. Dlaczego podział sekcji krytycznej na dwie mniejsze chronione osobnymi semaforami nie niweluje problemu konwojowania? Dlaczego blokada typu `mutex` rozwiązuje ten problem?

Zadanie 3. Użytkownik ustalając procedurę obsługi sygnału `sigaction(2)` może zażyczyć sobie by pewna grupa (jaka?) wywołań systemowych była **restartowalna**. Przeprowadź uczestników zajęć przez implementację `syscall`, po czym powiedz jak jądro (a) zwraca wynik wywołania oraz kod błędu «errno» (b) restartuje przerwane wywołanie?

Uwaga: Implementacja «sy_invoke» nie jest dla nas interesująca!

Zadanie 4. Na podstawie implementacji `trap_el0_sync` (AArch64) przypomnij uczestnikom zajęć jakie są główne zadania procedury obsługi pułapki. Należy również zajrzeć do `data_abort_handler`. Interesuje nas wyłącznie wariant obsługi pułapki, która wystąpiła w wyniku wykonania instrukcji w przestrzeni użytkownika.

Zwróć uwagę na wszystkie użycia procedury `trapsignal(9)` – do czego ona służy? Czy widzisz związek między jej parametrami, a parametrem `siginfo(2)` przekazywanym do procedury obsługi sygnału ustalonej przy pomocy `sigaction(2)`?

Uwaga: Implementacja «trapsignal» nie jest dla nas na chwilę obecną interesująca!

Zadanie 5. Załóżmy, że do procedury `copyout(9)` przekazano adres, pod który nie odwzorowano pamięci. Przeprowadź uczestników zajęć od początku procedury `copyout` aż do `copyio_fault_nopcb`, w którym zwróci ona «EFAULT». Odpowiednia procedura obsługi pułapki nazywa się `data_abort`.

Wskazówka: O tym co się dzieje można myśleć, jak o nielokalnych skokach przy użyciu procedury obsługi pułapki.

Zadanie 6. Przeprowadź uczestników zajęć przez procedurę dodającą i usuwającą wątek z **kolejki wątków uśpionych** (ang. *sleep queue*), odpowiednio: `sleepq_add` i `sleepq_remove_thread`. W szczególności rozważ przypadek kiedy wątek jest (a) jedynym (b) kolejnym uśpionym na danym **kanale oczekiwania**.

Zadanie 7. Odpowiedz na następujące pytania dotyczące `sleepqueue(9)`:

1. Implementacja `sleepqueue` nie wymaga, by z każdym możliwym `wchan` kojarzyć głowę kolejki uśpionych wątków. Czemu wystarczy, by jądro przypisało po jednym rekordzie `sleepqueue` do każdego wątku?
2. Zauważ, że między wykonaniem procedury «sleepq_add» i `sleepq_wait`, która usypia wątek, może dojść do wywłaszczenia wątku. Jak zatem implementacja unika problemu zagubionej pobudki?

Zadanie 8. Na podstawie [1] powiedz czym się różni **pamięć zadrutowana** (ang. *wired memory*) od **pamięci stronicowalnej** (ang. *pageable memory*)? Co jądro trzyma w pamięci stronicowalnej? Czemu nie można trzymać blokady `mutex(9)` (ani «MTX_DEF», ani «MTX_SPIN») robiąc dostępy do pamięci stronicowalnej?

Literatura

[1] „The Design and Implementation of the FreeBSD® Operating System”
Marshall Kirk McKusick, George V. Neville-Neil, Robert N.M. Watson;
Addison-Wesley Professional; 2nd edition, 2014

[2] „FreeBSD Device Drivers: A Guide for the Intrepid”
Joseph Kong; No Starch Press; 2012

[3] „UNIX Internals: The New Frontiers”
Uresh Vahalia; Prentice Hall; 1996