Struktura jąder systemów operacyjnych

Lista zadań nr 8

Na zajęcia 12 czerwca 2020

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

Zadanie 1. Rozważmy wielowątkowy proces, w którym jeden z wątków wywołał `exit(2)`. Na podstawie implementacji procedury `exit_lwps` wyjaśnij uczestnikom zajęć w jaki sposób jądro kończy wykonanie pozostałych wątków w procesie. Czy można wyznaczyć górne ograniczenie na czas wykonania wywołania «`exit`»? Gdzie jeszcze w jądrze wymagane jest użycie procedury «`exit_lwps`»?

Zadanie 2. Wszystkie wywołania systemowe służące do oczekiwania na zmianę stanu procesów, na przykład `waitpid(2)`, zbiegają do jednej procedury. Przeprowadź uczestników zajęć przez ciało `find_stopped_child`. Wyjaśnij sematykę flag `WNOHANG`, `WCONTINUED`, `WSTOPPED` i pokaż jak wpływają na logikę procedury. Czemu rodzic idzie spać na swoim «`p_waitcv`», jeśli procedura nie znajdzie żadnego dziecka o zmienionym stanie?

Zadanie 3. Jaką rolę pełnią w systemach `BSD` **aktywatory obrazów procesów** (ang. image activators)? Jakie jest zadanie procedur należących do `exec_elf64_execsw`? Przeprowadź uczestników przez proces przetwarzania pliku `ELF` w procedurze `exec_elf_makecmds`. Skoncentruj się na przeanalizowaniu scenariusza, w którym plik typu `ET_EXEC` posiada **interpreter**. Zauważ, że faktyczna konstrukcja nowej przestrzeni adresowej odbywa się w `execve_dovmcmds`, a struktury `exec_vmcmd` służą do opisu segmentów, które należy odwzorować przestrzeń adresową procesu.

Zadanie 4. Tablica `sigprop` przechowuje właściwości sygnałów uniksowych. Wyjaśnij różnice między definicjami właściwości dla: `SIGTRAP`, `SIGKILL`, `SIGSEGV`, `SIGSTOP`, `SIGCONT`. Rozważmy następujący scenariusz: sterownik terminala wczytał znak «`Ctrl+C`» i w wyniku jego przetwarzania zdecydował wysłać sygnał `SIGINT` do grupy pierwszoplanowej przy pomocy `kpgsignal(9)`. Wielowątkowy proces, do którego wysyłamy sygnał, nie jest śledzony, jest aktywny, nie blokuje sygnału i posiada zarejestrowaną procedurę obsługi. Przeprowadź uczestników zajęć przez procedurę `kpsignal2` wysyłającą sygnał do procesu.

Literatura

[1] „Modern Operating Systems”
Andrew S. Tanenbaum, Herbert Bos;
Pearson; 4th edition, 2015

[2] „Operating Systems Internals and Design Principles”
William Stallings; Pearson; 9th edition, 2018

[3] „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

[4] „Linux Kernel Development”
Robert Love; Addison-Wesley; 3rd edition, 2010

[5] „Solaris Internals: Solaris 10 and OpenSolaris Kernel Architecture”
Richard McDougall, Jim Mauro; Prentice Hall; 2nd edition, 2006

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

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

[8] „Mac OS X and iOS Internals: To the Apple’s Core”
Jonathan Levin; Wrox; 2012