Struktura jąder systemów operacyjnych

Lista zadań nr 10

Na zajęcia 24 czerwca 2020

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

Zadanie 1. Zaprezentuj przebieg obsługi wywołania systemowego `read(2)` na pliku reprezentowanym przez **v-węzeł**. Zacznij od implementacji `sys_read`, poprzez procedurę `dofileread(9)`, zestaw operacji `vnfileops(9)`, a kończąc na wywołaniu wirtualnego systemu plików `VOP_READ(9)`.

Zadanie 2. Zaprezentuj przebieg obsługi wywołania systemowego `write(2)` na pliku potoku. Od implementacji wywołania `sys_write` przejdź do procedury `dofilewrite` i zakończ w `pipe_write`. Po zamknięciu strony czytającej zapis do potoku powinien wysłać sygnał `SIGPIPE` do pisarza – wskaż miejsce, w którym następuje rozpatrywanie tego scenariusza. Główna pętla procedury `pipe_write` jest zależna od wartości pól struktury `uio`. Do czego służy procedura `uiomove(9)`? Wskaż miejsca w `pipe_write`, gdzie następuje zapis do bufora potoku. Gdzie rozpatrywane są atomowe zapisy do potoku? Kiedy wybudzamy procesy oczekujące na `select(2)` lub `poll(2)`?

Założenie: W trakcie czytania źródeł przyjmij, że makro «`PIPE_NODIRECT`» jest zdefiniowane.

W poniższych dwóch zadaniach rozważmy nasłuchiwanie na zdarzenie «`EVFILT_READ`» na potoku `pipe(2)` przy pomocy wywołania systemowego `kevent(2)`.

Zadanie 3. W tablicy `sys_kfilters` zdefiniowano zdarzenia, które można dodawać do `kqueue(2)`. Żeby z danym zdarzeniem powiązać **bilecik** (ang. knote) należy wywołać procedurę `kqueue_register`, która z kolei zawoła odpowiednią procedurę «`f_attach`». Zaprezentuj uczestnikom zajęć ścieżkę rejestracji zdarzenia aż do `pipe_kqfilter` włącznie.

Zadanie 4. W wyniku odczytu z potoku przy pomocy `pipe_read` wołana jest procedura `pipeselwakeup`. Prześledź jak dojdzie do wykonania procedury `filt_piperead`. Jakie informacje zostaną skojarzone z powiadomieniem o dostępności potoku do odczytu? Na podstawie `kevent(2)` powiedz jak odczytać te informacje w przestrzeni użytkownika.

Wskazówka: Po drodze zostanie wywołana procedura `knote` opisana w `knote(9)`.

Literatura

[1] „Magazines and Vmem: Extending the Slab Allocator to Many CPUs and Arbitrary Resources”[1]
Jeff Bonwick, Jonathan Adams

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

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

[4] „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

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

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

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

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

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

[1] `https://www.usenix.org/legacy/publications/library/proceedings/usenix01/full_papers/bonwick/bonwick.pdf`