Systemy operacyjne

Lista zadań nr 2

Na zajęcia 23–24 października 2018

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Tanenbaum (wydanie czwarte): 1.6, 1.7, 7.3, 10.2, 11.3
- Stallings (wydanie dziewiąte): 2.7 – 2.10

UWAGA! W trakcie prezentacji należy być gotowym do zdefiniowania pojęć oznaczonych **wytłuszczoną** czcionką.

Zadanie 1. Wywołania systemowe `fork`, `waitpid`, `lseek`, `rename`, `mmap` mogą zakończyć się błędem. W jaki sposób system operacyjny komunikuje programiście przyczynę niepowodzenia wywołania? Sprządź listę niepowtarzających się kodów błędów tak, by dla każdego wywołania w zrozumiały sposób wyjaśnić słuchaczom co najmniej dwie przyczyny niepowodzenia.

Wskazówka: Zapoznaj się z sekcją `ERRORS` odpowiednich stron rozdziału 2 podręcznika systemowego – użyj polecenia `man`.

Zadanie 2. W jakich przypadkach wywołanie stat(2)[1] mogłoby zakończyć się awarią systemu, gdyby jądro nie używało funkcji kopiujących[2] i nie sprawdzało poprawności argumentów? Czemu w trakcie przetwarzania wywołania systemowego należy skopiować dane pomiędzy **przestrzenią użytkownika** a **przestrzenią jądra**?

Zadanie 3. Które funkcje jądra systemu wykonują się w **górnej** (ang. *top half*), a które w **dolnej połówce** (ang. *bottom half*)? Jakie są konsekwencje zbyt długiego przebywania procesora w górnej połówce? Kiedy kod w dolnej połówce zostanie **wywłaszczony** (ang. *preempted*), a kiedy **wstrzymany** (ang. *suspended*)?

Wskazówka: Więcej o dolnej i górnej połówce można przeczytać w §3.1 książki *McKusick et al.* o `FreeBSD`.

UWAGA! Pojęcia górnej i dolnej połówki w systemie `Linux` mają odwrotne znaczenie niż w systemach `BSD`.

Zadanie 4. Wymień główne komponenty **monolitycznego jądra** Linuksa (§10.2). Czym charakteryzuje się jądro zorganizowane w **warstwy**? Wymień poważne wady architektury monolitycznej często przytaczane w literaturze. Które z nich można zniwelować dzięki użyciu **modułów jądra** oraz interfejsów typu `FUSE`[3]?

Zadanie 5. Jądro systemu `WinNT` jest łatwo **przenośne** (ang. *portable*) dzięki implementacji **warstwy abstrakcji sprzętu** (ang. *hardware abstraction layer*). Jakie zadania pełni `HAL` (§11.3)? Jakie korzyści daje `HAL` programistom implementującym **sterowniki urządzeń**?

Zadanie 6. Podaj motywacje stojące za wprowadzeniem systemów operacyjnych opartych na **mikrojądrach**. Ze względu na sposób komunikacji międzyprocesowej (ang. *Interprocess Communication*) w takich systemach każdy proces może pełnić rolę **klienta** lub **serwera**. Czy systemy z mikrojądrem to naturalni kandydaci na **rozproszone systemy operacyjne**?

Zadanie 7. Podaj przykłady **usług** i **sterowników** jądra monolitycznego, które są zrealizowane jako procesy użytkownika w systemie `Minix3`. Jaką rolę pełni **serwer reinkarnacji**? Na podstawie artykułu Reliability in MINIX 3[4] opowiedz jak architektura mikrojądra sprzyja **niezawodności** systemu.

Zadanie 8. System `WinNT` (§11.3) został pierwotnie zaprojektowany zgodnie z architekturą mikrojądra, ale w trakcie rozwoju zadecydowano o migracji do architektury **jądra hybrydowego**. Jednym z czynników było **mocne powiązanie** (ang. *tightly coupled*) komponentów – czemu jest to niepożądana cecha oprogramowania? Czym jest moduł **osobowości systemu operacyjnego** (ang. *OS personality*)? Jaką rolę pełni dodatek „Windows Subsystem for Linux” w systemie `Windows 10`?

---

[1] `http://netbsd.gw.com/cgi-bin/man-cgi?stat+2`
[2] `http://netbsd.gw.com/cgi-bin/man-cgi?copy+9`
[3] `http://lxr.free-electrons.com/source/Documentation/filesystems/fuse.txt`
[4] `http://wiki.minix3.org/doku.php?id=www:documentation:reliability`