TIN Zadanie 1

Początek realizacji: 21. 03 . 2020 Projekt wstępny: 04 . 04 . 2020; Koniec realizacji: .02. 06. 2020
punktacja: 0 - 50 p.; **Zespoły:** 4 os

Treść: Napisać program obsługujący uproszczoną wersję protokołu NFS (Network File System). Założenia:

- Należy zaimplementować serwer, bibliotekę kliencką (jako plik .a lub .so) oraz testowe programy klienckie realizujące funkcje operujące na zdalnych plikach (plikach zlokalizowanych na serwerze sieciowym)
- Zestaw funkcji do implementacji:
- `int` **`mynfs_open`**`(char *host, char *path, int oflag, int mode);`
  - `host` – nazwa DNS lub adres IP serwera
  - `path, oflag, mode` – jak w funkcji systemowej open() (dozwolone są modyfikacje lub ograniczenia)
  - zwraca: -1 gdy błąd, deskryptor pliku gdy operacja udana
  - należy zaimplementować co najmniej następujące tryby otwarcia pliku: `O_RDONLY, O_WRONLY, O_RDWR, O_APPEND, O_CREAT, O_EXCL, O_TRUNC`
- `int` **`mynfs_read`**`(), ` **`mynfs_write`**`(), ` **`mynfs_lseek`**`(), ` **`mynfs_close`**`()` - jak odpowiedniki systemowych funkcji: read(), write(), lseek(), close()
- `int` **`mynfs_unlink`**`(char *host, char *path)` – usunięcie pliku, parametry jak dla `mynfs_open()`
- `int` **`mynfs_opendir`**`( char *host, char *path)`
  - `host` – nazwa lub adres DNS serwera
  - `path` – jak funkcja opendir()
  - zwraca: -1 gdy błąd, deskryptor katalogu(!) gdy operacja udana
- `char *`**`mynfs_readdir`**`(int dirfd); int` **`mynfs_closedir`**`(int dirfd)`
  - dir_fd – deskryptor rekordu katalogowego zwrócony przez opendir
- Systemowe funckja readdir() zwraca za każdym razem strukturę dirent zawierającą rekordy opisujące kolejne pliki z katalogu, tu wystarczy, że funkcja zwróci samą nazwę, można jednak zastosować strukturę analogiczną do dirent, co będzie bardziej eleganckim rozwiązaniem.
- **`mynfs_closedir`**`(int dirfd)` – analogicznie do closedir()
- w razie wystapienia błędu należy kod informacyjny zapisywać w zmiennej globalnej `mynfs_error`
- Uwaga – należy przyjąć, że dozwolone jest wykonanie operacji `mynfs_open()` na katalogu, w trybie tylko do odczytu – wyłącznie w celu przekazania deskryptora do funkcji `mynfs_fstat()` (nie jest dozwolone zwykłe czytanie i inne operacje "plikowe" na katalogach)

Na co zwrócić uwagę:

- deskryptor zwracany przez mynfs_open() nie jest tutaj oczywiście deskryptorem plikowym – takim jak zwraca open(), creat(), itd.
- "Prawdziwy" protokół NFS zaimplementowany jest przy pomocy protokołów ONC RPC/XDR warstw 5 i 6, w tej uproszczonej implementacji nie należy bazować na RPC
- "Prawdziwe" NFS jest bezstanowe, tj. serwer nie przechowuje deskryptorów plików otwartych przez klientów, jednak w projekcie należy zastosowac rozwiązanie stanowe (jak dużo prostsze w implementacji)
- Należy wyspecyfikować (w proj. wstepnym), które z informacji dot. otwartego pliku są przechowywane po stronie serwera, a które po stronie klienta i jak lokalny deskryptor zwracany przez mynfs_open() mapuje się na zdalny plik. Istotne w tym konteście atrybuty pliku to: bieżąca pozycja i tryb otwarcia.
- Jak rozwiązana zostanie kwestia praw dostępu i autoryzacji użytkowników? Do rozważenia i opisania w proj. wstępnym (wskazówka – nie są wymagane skomplikowane rozwiązania)
- Co się stanie gdy usuniemy otwarty plik? (co się dzieje gdy usuwamy otwarty plik w systemie lokalnym? – łatwo można to sprawdzić...)
- Zalecana jest implementacja z wykorzystaniem protokołu TCP.

Warianty (każdyzespół realizuje jeden wskazany wariant)

- W1 – implementacja powinna bazować na UDP; można pominać funkcje open/read/close dir
- W2 – zaimplementować mechanizm blokad plików podobny do flock() (zob. man flock)
- W3 – przeprowadzić kompleksowe testy wydajnościowe
- W4 – zaimplementowac funkcję **`mynfs_fstat`**`(int mynfs_fd); -` pobiera atrybuty otwartego pliku - analogicznie do funkcji sysdtemowej fstat(); oraz **`mynfs_stat`**`(char *host, char *path); -` analogicznie do funkcji stat()

Uwaga: należy starannie zaprojektować protokół. Już w sprawozdaniu wstępnym należy szczegółowo go opisać

Instrukcje dot. realizacji projektu:

Terminy: 21.03.2020 Projekt wstępny: 04.04.2020; Koniec realizacji: 02.06.2020

Kwestie merytoryczne:

Sprawozdanie **wstępne** powinno zawierać:

1. Temat zadania, treść zadania, skład zespołu, data przekazania.
2. Interpretację treści zadania (tj. doprecyzowanie treści).
3. Krótki opis funkcjonalny – "black-box", najlepiej w punktach.
4. Opis i analizę poprawności stosowanych **protokołów komunikacyjnych** (wskazane - z rysunkami, np. zależności czasowych przy wymianie komunikatów, oraz postać/formaty komunikatów – np. w postaci tabelek lub rozpisanych w C struktur/obiektów).
5. Planowany podział na moduły i strukturę komunikacji między nimi (być może z rysunkiem), w tym koncepcję realizacji **współbieżności.**
6. Zarys koncepcji implementacji (język, biblioteki, narzędzia, etc.).

Nie należy opisywać kwestii znanych i omawianych na wykładzie, np. zasady funkcjonowania API gniazd, funkcji systemowych, standardowych narzędzi programistycznych, itp.

Projekt ostateczny powinien zawierać (6-15 stron):

1. To co projekt wstępny, jeśli potrzeba odpowienio zmodyfikowane i rozwinięte.
2. Pełen opis funkcjonalny "black-box".
3. Podział na moduły i strukturę komunikacji między nimi (silnie wskazany rysunek).
4. Opis najważniejszych rozwiązań funkcjonalnych wraz z uzasadnieniem (opis protokołów, struktur danych, kluczowych funkcji, itp.)
5. Szczegółowy opis interfejsu użytkownika.
6. Postać wszystkich plików konfiguracyjnych, logów, itp.
7. Opis wykorzystanych narzędzi, itp.
8. Opis testów i wyników testowania.

Uwagi dodatkowe:

- **Kodowanie**: język C/C++, środowisku Linux (lub inny Unix: BSD, ...)
- Testy (pokaz) powinny obejmować środowisko składajace się z co najmniej 3 węzłów, wskazane 4. Pokaz powinien obejmowac co najmniej 2 fizyczne komputery komunikujące się poprzez sieć (np. Wi-Fi w trybie ad-hoc lub poprzez ethernet – do dyspozyjci w czasie konsultacji jest przełącznik).
- Należy upewnić się wcześniej, że nie będzie problemów z nawiązaniem łączności między komputerami
- B. ważne jest precyzyjne opisanie obsługi **sytuacji wyjątkowych i reakcji na błędy.**
- B. ważne jest szczegółowe opisanie przeprowadzonych testów – UWAGA: testy nie mają na celu wykazania, że program **działa** poprawnie. Test ma na celu wykazanie, że program **nie działa** poprawnie!
- Punktacja: proj. wstępny: 10p; ogólna ocena realizacji projektu: 20 p.; sprowozdanie końcowe: jakość i kompletność: 10 p, jakość kodu z punktu widzenia inżynierii oprogramowania: 10 p.; w sumie: 50 p.

Uwaga: obecnie konsultacje odbywają się w tym samym terminie (wt 10-12) poprzez sesję Zoom (link będzie podany osobno). Wszystkie informacje o udziale w konsultacjach należy obecnie traktować jako dotyczące e-konsultacji. Informacje odnośnie dokumentacji drukowanej obecnie nie obowiazują, całość dokumentacji proszę przekazywać elektronicznie.

Kwestie organizacyjne:

- E-mail: g.blinowski@ii.pw.edu.pl; przypominam też o istnieniu ogólnodostępnej listy e-mail: tin.a@elka.pw.edu.pl
- Zasady korzystania z e-mail przy realizacji projektu: b. proszę zawsze podawać na początku tematu e-maila: „**TIN imię nazwisko ...**" *(dowolnie wybrany ale zawsze ten sam członek zespołu)*
- Konsultacje odbywają się zawsze we **wtorki w godz 10:05-12:00**; pok. 315; w wyjątkowych przypadkach mogą zostać przełożone na inny dzień.

- B. proszę w miarę możności stawiać się na konsultacjach w godz 10:05 – 11:00 (bez konieczności uzgodnienia) lub zasygnalizować chęć przybycia na konsultacje w godz 11:00-12:00, co najmniej dzień wcześniej mailem.
- Na konsultacje zawsze można zgłaszać się z dowolnymi pytaniami dot. realizacji projektu, zarówno organizacyjnymi jak i technicznymi. Zapraszam także do wysyłania maili w sprawach zarówno technicznych jak i organizacyjnych. **Konsultacje** w trakcie realizacji projektu nie wymagają obecności całego zespołu.
- Projekt wstępny proszę przekazać w wymaganym terminie (lub wcześniej) e-mailem na podany wcześniej adres podając w temacie: "TIN **Imię Nazwisko Projekt Wstępny**". Proszę nie przekazywać wydrukowanych sprawozdań wstępnych (oszczędzajmy środowisko ☺)
- Po zebraniu wszystkich (lub większości) projektów wstepnych opublikuję punktację oraz indywidualne uwagi.
- "Zdanie" projektu końcowego wymaga osobistego pojawienia się na konsultacjach; konieczne jest: (1) przeprowadzenie **pokazu** działania programu; (2) przekazanie **wydrukowanej** dokumentacji końcowej, Dopiero po wstępnym pozytywnym zaopiniowaniu projektu poproszę indywidualnie o (3) wysłanie e-mailem dokumentacji oraz źródeł do weryfikacj (lub poprawki). **Uwaga**: przez "zdaniem" projektu **nie należy** przekazywać mailem ani w innej postaci źródeł i/lub dokumentacji.
- **Pokaz** funkcjonowania musi odbywać się w obecności całego zespołu.
- Pokaz programu – odbywa się z reguly na własnym sprzecie studentów, w przypadku gdyby to bylo niemożliwe proszę o wcześniejszy kontakt.
- **Przekazanie źródeł:** źródła powinny być przekazane w postaci jednego pliku archiwalnego w formacie .zip, .tgz lub .tar.gz. Archiwum **nie może zawierać plików binarnych** (programów wykonywalnego, plików .o, plików roboczych repozytorium, itp.), dokumentację proszę wysłać jako drugi załącznik (nie powinna być częścią archiwum).
- **Uwaga** – z przyczyn: formalnych, organizacyjnych i technicznych nie akceptuję źródeł w postaci linku do zdalnego repozytorium.
- Nie jest możliwe wyłącznie e-mailowe zaliczenie projektu, jedyna dopuszczalna forma jest opisana wyżej.