Контрольна Робота

з предмету Операційні Системи

студента групи ІСзп 71
Бутузова О. В.

Варіант №4

Зміст

1 Операційна Система - FreeBSD 11.0

1.1 Історія створення

Історія створення серверної операційної системи FreeBSD бере свій початок ще у Bell Labs з проекту Research Unix (спецверсій UNIX що використоувались на різноманітних дослідницьких компютерах), що також дав початок VAX та іншим UNIX системам 70-80 років. Якщо конкретніше - то FreeBSD базується на оперційні системі розроблені та розповюджувані дослідницькою групою компюторних систем університету Berkeley (CSRG) що була похідною від тогочасних UNIX систем та мала назву Berkeley Software Distribution або BSD. З точки зору історій BSD це підсімейство Unix що дало початок розробкі не тільки FreeBSD - а і OpenBSD, NetBSD, DragonflyBSD та Darwin (більш відомому наразі як MacOS або OS X) .

Першим з публічних резізів BSD була Net-1 і оскільки вона була частково базована на коді що був отриманий з AT&T (і була власністю цієї компанії) для використання Net-1 потрібно було ще отримати ліцензію від правовласників, що спричиняло незручності як легального так і фінансового плану для розвитку університетського проекту BSD. Ще до релізу Net-1 студенти та викладачі почали переробляти ліцензовані AT&T компоненти (TCP/IP, віртуальну память, файлову систему тощо) і до моменту реліза Net-2 у системному ядрі залишалось 6 файлів базованих повністю на програмному коді з AT&T. Один з провідних розробників BSD - Кейт Бостікґ запропонував зробити реліз Net-2 без цих шістьох файлів щоб позбутись обмежень ліцензування від AT&T, Net-2 не була повноцінною операційною системою без них, але вже не потребувала ліцензування від правовласників.

Історія вже саме FreeBSD почанається у 1992 році, коли Вільям та Лін Джоліц написало заміну цим шістьом файлам і невдовзі була представлений першоджерельний код написаний на мові програмування C, що міг бути скомпільований у систему, що могла завантажуватись і працювати на 386 процесорах компанії Intel - вони назвали свій проет 386BSD. Через те що розробка 386BSD

не була дуже успішною (подружжя Джоліц були зосереджені і на інших власних проектах) - автори патчів до 386BSD вирішили відгалузитись від основного коду застувавши власний реліз, який би був більш сфокусований на не настільки технічно обізнаних користувачах і міг би запропонувати краще підтримку користувачам.

Компанія видавець Wallnut Creek CDRom - у грудні 1993 року зробила перший реліз FreeBSD. Він відбувся на базі FTP серверів компаній так і на CDROM новому форматі компактних дисків. Компанія займалась в подальшому виданням літератури про ОС, проведенням конференцій тощо і підтримкю FreeBSD на всіх фронтах. Наразі компанія переіменована в iXsystems і є авторами одного з найпопулярніх продуктів базованих на FreeBSD під назвою FreeNAS. Майже одразу після релізу AT&T подала до суду позов "про порушення авторського права відповідачами були NetBSD та FreeBSD, позов був погоджений сторонами конфлікту на невідомих умовах (деталі залишаються невідомими і по сьогодні), але як результат FreeBSD та NetBSD продовжили свій розвиток вже без легальних проблем зі сторони AT&T.

З 1993 року по 2017 видано 12 версій (11 стабільних і 12та над якою ведеться робота зараз) операційної стистеми FreeBSD, вона зазанала як божевільної популярності, коли у 2005 була дефакто стандарнтною серверною операційною системою для корпоративного серктору, так і "тяжких частів"що ознаменувались витісненням FreeBSD з корпоративного сектору конкуретом - CoreOS. FreeBSD дала дорогу багатьом іншим операційним системам - найпопулярнішими з яких лишаються операційні системи сімейства корпорації Apple (в яку вони потрапили разом з поверненням Стіва Джобса який приніс ОС NextStep як базу для OS X що по суті врятувала Apple у кінці 90тих). FreeBSD наразі позиціонує себе як середовище, не тільки для корпоративних серверів, так і як система для приватного сектору, такою FreeBSD займає частинку ринку для вбудованих систем (як операційна система пристроїв з обмеженою функціональністю).

1.2 Переваги FreeBSD

Значними перевагами FreeBSD у порівнянні з іншими UNIX/Linux системами є такі особливості:

- Гарна підтримка мережевого обладнання і оптимізація для 100gbps мереж: З оптимізованими драйверами пристроїв від усіх основним виробників 100gbps мережевого обладнання FreeBSD (починаючи з версії 7.0) отриала загальну отпимізацію мережевого стеку для високонавантажених систем, включаючи автомаштабування сокетних буферів, TCP Segment Offload (TSO), Large Receive Offload (LRO), пряме управління мережеивим стеком і балансування навантаження при обробці TCP/IP на системах з кількома CPU з підтримкою 100gbps або при одночасному використанні кількох мережевих інтерфейсів. Повна підтримка пристроїв наступних вироників Chelsio, Intel, Myricom и Neterion. За допомогою драйверів від Yandex стала доступна одночасна обробка TCP/IP пакетів.

- SMPng: механізм синхронізації в ядрі що забезпечує лінійне маштабування на більш ніж 8 ядрах CPU. В усунені великі блокування (Giant Lock) і поснітю прибрані з рівнів абстракції технологій збереження CAM (Common Access MEthod Layes) і клієнтів NFS, виконаний перехід на більш диференційовану (fine-grained) винзронізацію у мережеві підсистемі. Оптимізовані планувальники ядра та примітиви синхронізації, опціональний планувальник ULE забезпечує привязку процесів до потоків CPU і черги запуска для кожного з CPU для зменшення видатків і відвищення ефективності роботи кеша. Використання бібліотеки libthr що реалізує 1:1 багатопоточність.

- SCTP: FreeBSD має увімкнуту по замовчуванню реалізацію нового протоколу передачі з управлінням потоком IETF - Stream Control Transmission Protocol (SCTP), створеної для пітримки VoIP телефонії, телекоммунікацій та інших аплікацій з чіткими вимогами до надійності і передачею зі знімнною якостюі з такими особливостями як багато-дорожна (multi-path) доставка, відказонадійність (fail-over) і багато поточність (multi-streaming).

- Wireless: постійно удосконалюється стек технологій що дозволяють вико-

ристовувати потужні мережеві (бездротові технологій) від Atheros, і інших виробників бездротового мережевого таких як Ralink, Intel и ZyDAS. Підримка WPA, WPA2, фонового сканування.

- Дозволяє використання Sun ZFS (а також OpenZFS): сучасної файлової системи, що має своїми особливостями просте адміністрування, транзакційну семантику і безперервну цілістність даних. Від самовідновлення до вбудованої компресії, підтримки raid, знімків системи і управління томами ZFS що дозволяє системним адміністраторам прості інструменти управління надвеликими масивами данних.

- Реалізацію для багатьох архітектур серед яких: 386/486/586/686 IA-32, amd64, x86-64, Ultra SPARC, PowerPC, ARM, MIPS

- Обєднаний кеш віртуальної памяті і буферів файлових систем оптимізує розподілення памяті та дискового кешу. що використовується программами, в результаті чого програми отримують високоефективний доступ до файлової системи динамічно, без налаштувань розмінів кешу системним адмінастратором.

- Модулі сумістності, що дозволяють программам скомпільованим під інші операційні системи такі як (Linux, SCO UNIX, System V) виконуватись в середовищі FreeBSD.

- Підтримка IP Security (IPsec) та IPv6

- Пріорітетність задач - що дозволяє високо приорітетним задача витісняти низько приорітетні з ланцюжка виконання в ядрі. Сюди можна включати висоефективну багатопоточний мережевий стек та багатопоточну віртуальну память.

- DTrace для налагодженя та відладки ядра "вживу".

- Віртуалізація мережі. Контенейр ("vimage") було імплементовано на рівні ядра що дозволило йому (ядру FreeBSD) справлятись з багатьма незалежними інстансами мережевого стану, що в свою чергу дозволило створювати доволі складні віртуальні мережеві топології напрякладі подібних

що створюються пакетом віртуалізації Docker/Moby. Окрім того можливе використання Docker напряму (експеременталька підтримка).

- Коллекція портів - вже скомпільованого програмного забезпечення що може бути запущене на FreeBSD.
- Jails - легковісна альтернатива віртуалізації що дозволяє обмежити процес до простору імен для яких дозволено лише файловий/мережевий доступ. Jails довзоляє сворювати ієрархію навіть внутрі самого себе (що в свою чаргу дохзволяє обмежувати вже обмежені процеси).
- Брендмауер встановлений по замовчуванню.

1.3 Ядро системи

Переваги FreeBSD не обмежуються цим списком, систему можна конфігурувати на власний розсуд/потреба/можливості. Це можна зробити з допомогою налаштування ядра системи, що зазвичає є першим кроком у конфігурації freebsd системи взагалі. Ядро, або kernel, системи FreeBSD можна налаштовувати динамічно або вживу. За необхідністю можна змінювати більшість аспектів швидкодії системи, слід зауважити що певні елементи ядра не можуть бути змінені безпосередньо під час роботи системи, або змінені взагалі.

Рекомендується збирати власне ядро для викорастання на перних конфігураціях, для цього варто лише переіменовувати директорію "kernel"з ядром що знаходиться у директорій /boot/, доречі всі інші директорій поза директорії з ядром називають userland. Але повернемось до ядра, точніше на самий початок що воно таке і навіщо воно пострібно в FreeBSD (і інших операційних системах).

Так ось ядро по суті є інтерфейсом між апаратною і програмною частиною. Ядро краще знає як і куди записувати данні, як працювати в мережі, як перетворюбвати сукупність бітів у tiff файлі і яким чином посилати ці данні на друк чи на вивід на екран. Коли программа дає запит на виділення їй ресурсів дял обчислень саме ядро обслуговує цей запит та виділяє ресурси що були запрошені.

1.4 Кастомізація ядра

В спрощеному вигляді ядро це программи та файли конфігурацій що як вже було сказано знаходяться в директорії /boot/kernel, інтерфейсом що використовувати з командної стрічки для перегляду або зміни опцій є команда sysctl. Загалом ядро системи містить наступні розділи

Sysctl	Призначення
kern	Основні функції ядра
vm	Підсистема віртуальної памяті
vfs	Фаилові системи
net	Мережеві функції
debug	Інформація відладки
hw	Апарація щодо апаратної частини
user	Інформація щодо простору user
p1003_1b	Параметр керуючі характеристиками POSIXa
compat	Сумістність ядра з програмним забепеченням інших операційний систем
security	Параметры обеспечения безопасности
dev	Інфомрація щодо драйверів пристроїв

Отримати розгорнуту довідку щодо кожного з розділі можна набравши команду sysctl з ключем відповідного розділу, наприклад sysctl hw (видасть інфомарцію про апаратне забезпечення на сервері).

```
hw.ncpu: 24
hw.byteorder: 1234
hw.memsize: 103079215104
hw.activecpu: 24
hw.targettype:
hw.physicalcpu: 12
hw.physicalcpu_max: 12
```

sysctl довзоляє не тільки виводити але і змінювати певні системні параметри

ядра за допомогою оператора еквівалентності (=) наприклад:

```
> sysctl net.inet.ip.ttl=128
net.inet.ip.ttl: 64 > 128
```

Можливо також отримувати довідку щодо того чи іншого значення (якщо така опція доступна для того чи іншого налаштування) за допомогою ключа d". Значення (реальне значення чи опис) багатьох опцій ядра можна знайти лише в перешоджерельному коді (за умови якщо воно там є) тому запит на опис не завжди спрацьовує. Опцій ядра також можна задавачи через конфігураційний файл ядра (kernel) що має свій власний формат. З вишенаписаного можна здогадатись що ядро FreeBSD є монолітним з динамічно-завантажуваними модулями які користувач може налаштувати "під себе".

1.5 Користувацький ринок та графічні оболонки

Окрім сектора серверів, FreeBSD також позиціонує себе як система для настільних компютерів та ноутбуків, тобто для користувацького сектору. Якщо на серверах викоритовують одну з оболонок (будьто csh або tsch) то більшості користувачів потібна графічна система, роль якої у FreeBSD можутуь відігравати кілька різних оболочок що підтримуються різними командами (наприклад команда GNOME підтримує крім своєї оболонки щей mate та Cinnamone). Наразі доступні xfce, gnome, kde plasma, mate, cinamone, i3 та інші.

1.6 Ринок вбудованих систем

Окрім серверного та користувацького сектору, FreeBSD можливо використовувати на вбудованих системах (напирклад на роутерах або міні компюторах). Можна навіть стверджувати що у проекту FreeBSD завдяки цьому сектору відбувається другий подих, оскільки неймовірно популярності набуває таке якище як IoT або Internet of Things, де надійні і відмовостійкі системи потрібн як ніколи. FreeBSD пропонує вже прекомпільовані образи системи що їх можна

скачати з домашньої сторінки FreeBSD скачати вже готовими для наступних плат розробки: Raspberry PI, Banana, Eagle, CUBIE та ін.

1.7 Версія 11

Версія 11 Операційної системи FreeBSD вийшла у жовтні 2016 року і несло в собі численні зміни в порівнянно з попередньою версією. Зміни в основному стосувались виплавлень в ядрі, додачею нових модулей ядра що забезпечували роботу нових пристроїв, виправленням коду для покарщеної компіляції за стандартом C++11 (стандарт 2011 року), оновлення чиленниз бібліотек. оновленням ряду функціональностей программи bsdinstall, покращеною підтримкою віртуалізації і покращенням роботи на архітектурі arm (в основному iot пристрої).

1.8 Висновки

FreeBSD хоч і не займає наразі лідуючі позиції на корпоративному ринку серверних OS, але списувати з рахунків таку систему вкрай ще рано і розробка триває безперестанно.

Використані джерела

1. Darwin/Mac OS X: The Fifth BSD
   https://networking.ringofsaturn.com/Unix/bsd.php
2. Features of FreeBSD
   https://www.freebsd.org/features.html
3. FreeBSD's SMPng
   http://www.onlamp.com/pub/a/bsd/2005/01/20/smpng.html
4. Доступный UNIX: Linux, FreeBSD, DragonFlyBSD, NetBSD, OpenBSD
   https://books.google.com.ua/books?id=JjOApZ2m0JUC
5. Лукас М. FreeBSD. Подробное руководство
   https://www.michaelwlucas.com/os/af2e

6. Download FreeBSD
   https://www.freebsd.org/where.html
7. The Top 10 Things to Know About the Internet of Things (IoT) as a PCB Designer
   https://www.autodesk.com/products/eagle/blog/top-10-things-about-iot-pcb-designer/
8. FreeBSD 11.0-RELEASE Release Notes
   https://www.freebsd.org/releases/11.0R/relnotes.html
9. Мифы о FreeBSD
   https://habrahabr.ru/company/ua-hosting/blog/306804/
10. Сервер на стероидах: FreeBSD, nginx, MySQL, PostgreSQL, PHP и многое другое
    https://habrahabr.ru/post/70167/

2 Оболонка - tsch

2.1 Екскурс в історію

tcsh або TENEX C Shell це Unix оболонка створена Кеном Грігом. Вона бере ім'я від операційної системи TENEX (TEN EXtended) що була розроблена на мові програмування LIST для мейнфрейму DEC PDP-10. Якщо для UNIX використовувались короткі команди наприклад ls (для виводу каталога), то для TENEX аналогом такої команди була DIRECTORY (OF FILES) де власне командою було DIRECTORY, а (OF FILES) було "шумом"або додатковою опцією ("щоб було зрозуміліше"). Ясно що такі команди було вводити трошка складніше і TENNEX мав встроєну систему автодоповнення, опції що багато UNIX систем не мали.

Робота над tcsh почалась у версні 1975 і закінчилась фінально злиттям проекту з csh (C shell) у вересні 1983 року, через місяць першоджерельні файли були надіслані до usernet'івської групи net.sources.

tcsh є оболонкою по замовчуванню в відкритих операційних системах FreeBSD та також пропрієтарних, наприклад UNIX подібних системах компанії IBM - OS/390 і z/OS, та FreeBSD похідхних NextComputer та Apple (перші версії OS X використовували tcsh як основну оболонку, але починаючи з версії 10.3 оболонкою по замовчуванню є bash).

2.2 Основні функції

Як і у інших оболонок основною функцією tcsh є:

- Запуск команд або утиліт що запитує система.
- Написання сценаріїв оболонки використовуючи власну мову програмування.
- Запуск сценаріїв оболонки та програм інтерактивно або у фоні.

Основними можливостями tcsh в порівнянні з іншими оболонками є:

- Історія вже запущених команд
- Редагування командного рядка з підтримкою стилей від vi або emacs
- розширений механізм нафігації по каталогах
- Авто-доповнення як файлів так і імен змінних що користувач може друкувати в командному рядку.
- Аліаси до аргументів що можуть бути перенаправлені до відповідних команд.
- Управління задачами
- команда where (аналогічна which) що показувала усі локейшени шуканого обєкту.

2.3 Спеціальні можливості

Саме нявність автодоповнення що дозволяв вводити довгі рядки команд забезпечила популярність tcsh на довгі роки, і саме до цієї особливості зводиться використання tcsh більшостю користувачів. Слід також зосередити увагу на відмінностях в роботі в порівнянні з найпоширенішою на сьогодні оболонкою bash.

- alias/unalias працють по іншому в tcsh ніж в bash, замість оператора еквівалентності (=) використовується табуляція або пробіли
- Не зручний варіант написання сценарії: що виражається в неможливості використання функцій деклароваий напряму в файлі сценарію.
- вбудована команда history що виводить останні команди (і працює трошки відмінно від аналогічної команди у bash).
- history expansion - схоже за схемою на bash аналог, але дозволяє більш гнучко визивати команди х історії наприклад !123 команди в списку історії номер 123 (123 з кінця), а !?cat? виклче останнію команду що містинm

"cat". Слід не оминути і модифікатори, що дозволяють змінювати параметри розширення історій в процесі виконання.

- set що дозволялал змінювати певні змінні, такі як history (кількість записів історії що ви виведемо) і savehist (кількість записів у файлі .history) та інших... (а такою команди unset що обнуляла данні значення)
- Спеціальні псевдоніми - виконання команд при певних умовах (наприклад автоматичне виконання команд що є псевонімами до cwdcmd перед тим як робочий каталог буде змінено)
- Управління задачами - працює аналогічно окрім моменту коли tcsh виводить усі PID що належать виконуваному процесу.
- перенаправлення stderr працює інакше - через >&
- Автозавершення імен файлів (просто написавши початок імени файлу/папки і натиснувши TAB).
- Автозавершення команд (просто написавши початок команди і натиснувши TAB CTRL+D)
- Редагування командної строки - про яке я казав раніше являє собою просте переназначення функцій клавіш і навішування на них певних команд.
- Арифметичне розширення (розкривається чреез '@') дозволяє працювати з змінними що збережені як строки як з числами.
- використання RAA (що я вляють собою значення заключені в дужки).

2.4 Недоліки

Наразі tcsh як і FreeBSD пережеває не найкращі часи: функції що довгий час були фішкою саме tcsh вже давно так чи інакше імплементовано в bash або інші оболонки. Програмування в tcsh не є приємною чи безпечною задачею, перенаправлення потоків працює не зовсім зрозуміло для новачка порівнюю з простотою bash.

2.5 Висновки

За довгі роки tcsh так і не набула критичної маси користувачів не дивлячись на численні новаторські функціональності і продовжує використовуватись практично лише серед FreeBSD адміністраторів.

Використані джерела

1. tcsh (C Shell) Kit Support Guide
   ftp://ftp.software.ibm.com/s390/zos/unix/ftp/tcsh.pdf
2. TOPS-20
   https://en.wikipedia.org/wiki/TOPS-20TENEX
3. М. Собель - Linux. Адміністрування та системне програмування. 2-ге вид.
   https://books.google.com.ua/books?id=YfnNlmUFwsMC
4. Csh Programming Considered Harmful
   http://www.faqs.org/faqs/unix-faq/shell/csh-whynot/
5. Командная оболочка tcsh
   http://citkit.ru/articles/1107/

3 Системний виклик (syscall) - write

Syscall - в програмуванні це звернення программи до ядра операційної системи для виконання будьякої операції.

3.1 Що таке системний виклик write

Системний виклик write є одним з базових викликів що проваджується ядром UNIX подібної операційної системи. Основною його функцією є запис декларованого користувачем буферу певним розміров в певний пристрій. Пристроєм при цьому може бути як файл, так і stdout (поток виводу) або stderr (поток помилок), цей аргумент має задаватись цілим числом (зокрема в мові програмування C типом int). В POSIX існує три стандартних файлових дискриптора відповідно до трьох стандартних потоків що повязані за кожним процесом.

3.2 Дискриптори файлу

Ціле Фисло	Опис	Символічна константа	Потік
0	Стандартний ввід	STDIN_FILENO	stdin
1	Стандартний вивід	STDOUT_FILENO	stdout
2	Стандартний вивід помилок	STDERR_FILENO	stderr

Окрім того стандартних файлових дескрипторів - декскриптор може бути створено за допомогою інших системних викликів - таких як:

- open
- creat
- socket
- accept
- socketpair
- pipe

- opendir

Кожен з яких може працювати з певними примітивами і рівнями абстракції (наприклад з сокетами або файлами).

3.3 Інтерфейс write

Дані що будуть записані (допустимо для прикладу - що це якийсь текст) задекларований вказівником (pointer'ом) і розміром цього тексту в байтах (по іншому в мові програмування C не можливо передати масив данних, лише як масив і його довжину окремо), останній аргумент ще трактують як число байт з масиву buf що треба записати.

Інтерфейс write стандартизовано специфікаціє POSIX і описано в багатьох джерелах. Прототип функції виглядає наступним чином (ssize_t декларується в бібліотеці stddef.h)

ssize_t write(int fd, const void *buf, size_t nbytes);

Як видно з прототипу інтерфейсу системний виклик такою повертає деяке число, це число байтів що були 'кудись' записано. У разі помилки повернеться unsigned int. Нижче наданий перелік можливих помилок в які можна попасти з файловими дескрипторами.

3.4 Типові очікувані помилки при використанні write

N	Код	Опис
4	EINTR	Системний виклик було перервано.
5	EIO	Низько-рівнева помилка, повязана з чит/зап апаратного заб-ня.
9	EBADF	Дескриптор не валідний або файл фідкритий в режимі читання.
13	EACCES	Користувач не має права запису до файлу.
14	EFAULT	Адреса вказана не правильно.
22	EINVAL	Аргументи передані функцією не можуть бути валідовані.
27	EFBIG	Перевищено розмір допустимого розміру буферу.
28	ENOSPC	Недостатньо місця для запису у файл.
32	EPIPE	Пробелема передачі потоку або файл не готовий до передачі.

3.5 Імплементація, використання та тестування

Компіляція программи
gcc write.c -o writer && chmod 0777 ./writer

Запуск та отримання проміжних варіантів
cat sample.txt | xargs ./writer result.txt && chmod +rw result.txt

Перевірка результатів
md5 result.txt && md5 sample.txt

> Через неможливість використання кирилиці у пакеті LaTeX listnings коментарі подано у вигляді транслітерації.

Код ніжче наведенної программи намагається створити файл якщо файл не існує, або відкриває файл якщо він існує і пише до цьогофайлу усі аргументи що зустрічаються після нього. Тобто запит до программи

./writer test.txt This is test!

завершиться записом файлу test.txt з змістом "This is test!"

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

// Cia Programma sprymae pershym argumentom ima failu,
// i pishe do niogo use show bude peredano piznishe.

int main (int arg_count, char *arg_verbs[]) {

    if ( arg_count < 3 ) {
        printf("./write filename.txt any extra data\n");
        return 1;
    }

    // inicializacia failovogo descriptory
    int fd;

    // Iakcho faily ne isnue to mi iogo stvorimo
    // abo vidkriyemo iakcho vin dostupni
    if ( ! access( arg_verbs[1], F_OK ) ) {
      fd = open( arg_verbs[1], O_WRONLY );
    } else {
      fd = creat( arg_verbs[1], O_WRONLY ) ;
    }

    // U vypadku pomilky
    // vyidemo z programmy
    if ( fd == -1 ) {
        perror(arg_verbs[1]);
        return EXIT_FAILURE;
    }

    // Using
    char* end = " ";

    for (int counter = 2; counter < arg_count; counter++ ) {
        write(fd, arg_verbs[counter], strlen(arg_verbs[counter]));

        // rozdiluvach
        if ( counter != ( arg_count - 1 ) ){
          write(fd, end, 1);
        }
    }

    // Final endl character.
    end = "\n";
    write(fd, end, 1);
    close(fd);

    return 0;
}
```

Використані джерела

1. Searchable Linux Syscall Table for x86 and x86_64
   https://filippo.io/linux-syscall-table/
2. Linux syscalls list
   https://syscalls.kernelgrok.com/
3. The Definitive Guide to Linux System Calls
   https://blog.packagecloud.io/eng/2016/04/05/the-definitive-guide-to-linux-system-calls/
4. Linux Programmer's Manual - Linux system calls
   http://man7.org/linux/man-pages/man2/syscalls.2.html
5. Learn C Programming
   https://www.programiz.com/c-programming
6. C Programming and C++ Programming
   https://www.cprogramming.com/