РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: програмний додаток для автоматизації завантаження відеоконтенту в мережі Інтернет.

Мета кваліфікаційної роботи: розробка програмного додатку для завантаження відео з різних ресурсів та його перегляду онлайн без використання браузеру.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано платформу для розробки, виконано проєктування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення визначається повноцінною розробленою програмою Cool Video Downloader, що виконує функції менеджера завантажень та відеопрогравача і може стати в нагоді широкому спектру користувачів. Також дана програма підтримує скачування з популярної мережі для перегляду відео Anilibria, Dark-libria та Jutsu. Вона дозволяє змінювати озвучення та отримувати зображення хорошої якості, завдяки чому перегляд відео буде більш корисним та приємним, чого іноді важко досягнути при використанні браузеру.

Актуальність даного програмного продукту випливає з того, що часті перебої з мережею Інтернет та відсутній доступ до неї при перегляді відео, а також велика кількість реклами заважає зручному та комфортному перегляду контенту. Таким чином, повнофункціональна програма для завантаження та перегляду відео без використання браузеру допоможе уникнути цих проблем.

Список ключових слів: ВІДЕФАЙЛИ, BASH-ОБОЛОНКА, МЕДІАПЛЕЄР, БРАУЗЕР, ПРОГРАМА, ТРЕКЕР.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: a software application for automating the download of video content on the Internet.

The purpose of the qualification work: development of a software application for downloading video from various resources and watching it online without using a browser.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

In the second section the analysis of existing solutions is performed, the platform for development is chosen, the program is designed and developed, the algorithm and structure of program operation are described, input and output data are defined, characteristics of technical means parameters are given, call and program loading are described.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical value is determined by the full-fledged program Cool Video Downloader, which acts as a download manager and video player and can be useful to a wide range of users. The program also supports downloads from the popular Anilibria, Dark-libria and Jutsu videos. It allows you to change the sound and get good quality images, making video viewing more useful and enjoyable, which is sometimes difficult to achieve when using a browser.

The relevance of this software product stems from the fact that frequent interruptions to the Internet and lack of access to it when watching videos, as well as a large amount of advertising interferes with the convenient and comfortable viewing of content. Therefore, a full-featured program for downloading and watching videos without using a browser will help avoid these problems.

Keywords: VIDEFILES, BASH-SHELL, MEDIA PLAYER, BROWSER, PROGRAM, TRACKER.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

TCP – Transmission Control Protocol, протокол керування передачею;

TCP/IP – Transmission Control Protocol / Internet Protocol (протокол управління передачею / міжмережевий протокол;

TFTP – Trivial File Transfer Protocol, тривіальний протокол передачі файлів;

UDP – User Datagram Protocol, протокол дейтаграм користувача;

URL – Uniform Resource Locator, єдиний вказівник на ресурс;

FTP – File Transfer Protocol, протокол передачі файлів.

ЗМІСТ

ВСТУП

Поява доступу до мережі Інтернет та розвиток технологій значно змінили життя сучасного суспільства. Смартфони, ноутбуки, комп’ютери тепер є у кожному домі та здатні замінити безліч інших пристроїв. За допомогою них люди працюють та навчаються, проводять свій вільний час за читанням, іграми чи переглядом відео. Саме тому розробники змагаються у створенні найбільш оптимізованих програм, завдяки яким користувач буде витрачати якомога менше часу на пошук та завантаження інформації, натомість використовуючи його на більш корисні або приємні заняття.

Актуальність роботи випливає з того, що часто користувачі жаліються на те, що в них є перебої з мережею Інтернет, відсутній доступ до неї, при перегляді відео з’являється велика кількість реклами чи потрібний ресурс просто недоступний з тих чи інших причин.

Згідно статистики YouTube [13], відеоконтент для розваги чи навчання щодня споживають біля 1,9 мільярдів користувачів, 62% компаній використовують дану платформу для публікації власних відео і більше 70% часу перегляду приходиться на мобільні пристрої. Ці дані показують масштаби перегляду відео в світі, а отже підтверджують потребу в повнофункціональній багатозадачній програмі для завантаження та перегляду онлайн відео без використання браузеру.

Таким чином, повнофункціональна програма для завантаження та перегляду відео без використання браузеру допоможе уникнути цих проблем. Також дана програма підтримує скачування з популярної мережі для перегляду відео Anilibria, Торрент трекерів, та інших ресурсів. Вона дозволяє змінювати озвучення та отримувати зображення хорошої якості, завдяки чому перегляд відео буде ще більш корисним та приємним, чого іноді важко досягнути при використанні браузеру.

Отже, мета роботи полягає в створенні програми для завантаження відео з різних ресурсів та перегляду онлайн без використання браузеру.

Для досягнення мети було поставлено наступні завдання:

- дослідити особливості функціонування схожих програм;
- визначити найкращі способи оптимізації для програми;
- визначити специфіку роботи мови програмування Bash;
- змоделювати програмну платформу роботи;
- розробити програму;
- протестувати розроблений додаток для скачування та перегляду відео і порівняти його з іншими програмами шляхом визначення її переваг та недоліків.

В результаті виконання даної кваліфікаційної роботи створено програмний додаток Cool Video Downloader – повнофункціональна багатозадачна програма для завантаження й перегляду відео без використання браузеру.

Практичне значення виконання роботи визначається повноцінною розробленою програмою Cool Video Downloader, що виконує функції менеджера завантажень та відеопрогравача і може стати в нагоді широкому спектру користувачів.

Дана робота має прикладний характер, оскільки її можна ефективно використовувати в повсякденному житті.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Сьогодні користувачі світової мережі використовують високошвидкісні і широкосмугові інтернет-канали, що дозволяють без проблем переглядати відео в режимі онлайн. Але іноді все-таки виникають ситуації, коли необхідно зберегти той чи інший відеофайл в домашній архів на комп'ютері. На жаль, завантажувати відео через веб-браузер на високій швидкості і без збоїв з'єднання вдається не завжди і не на всіх сайтах. Тому найкращим вирішенням цієї проблеми є спеціальні програми для скачування відео з інтернету. Оскільки метою даної роботи було створення такої програми, то з’явилася необхідність в порівнянні її з аналогами. Серед найбільш популярних аналогів можна виділити наступні. [12]

1.1.1. Video Dowloader від Jihosoft

Jihosoft Video Dowloader - це одна з найбільш популярних програм в цій категорії. З її допомогою можна не тільки завантажувати окремі відео, а навіть плейлисти і цілі канали. Також доступна конвертація відео в інші формати, субтитри і оригінальні аудіодорожки. Підтримується якість до 8 К.

ihosoft 4K Video Downloader - це потужний завантажувач і конвертер відео з YouTube, спеціально розроблений, щоб допомогти користувачам завантажувати відео з YouTube і конвертувати відео з YouTube в MP3. Він об'єднує кілька функцій в одну і надає повний спектр послуг. Використовуючи його, ви можете швидко завантажувати улюблені відео і фільми для перегляду в автономному режимі без реклами і буферів (рис. 1.1).

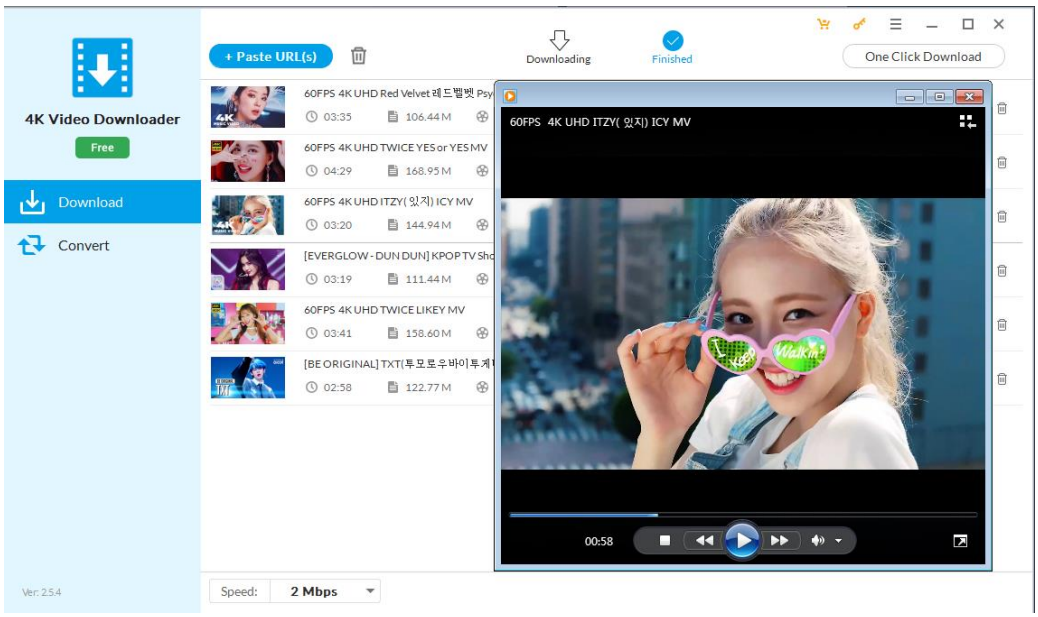


Рис. 1.1. ihosoft 4K Video Downloader

1.1.2. SaveFrom.Net

SaveFrom.Net - це зручне розширення для браузера з можливістю скачування файлів з більшості сайтів. Розширення доступно для всіх популярних браузерів (рис. 1.2).

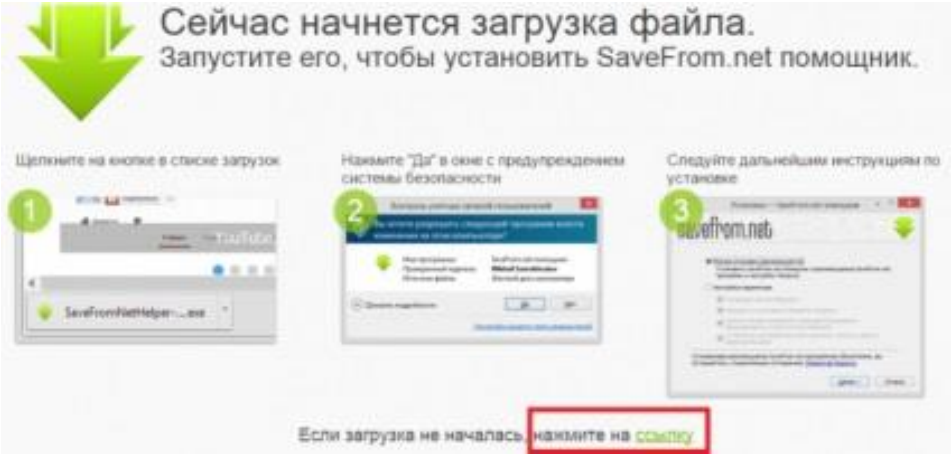


Рис. 1.2. SaveFrom.Net

SaveFrom.net робить процес завантаження з мережі зручним та простим. За його допомогою можна завантажувати аудіо, відео та інші типи файлів зі різних сайтів та соціальних мереж: youtube.com, vk.com та інших.

Переваги сервісу SaveFrom.net»:

1. Швидкий та зручний доступ з поля адреси.

2. Завантаження відео прямо з будь-якої інтернет сторінки.

3. Отримання списку посилань для завантаження, у випадку якщо кілька відеофайлів наявні на одній веб-сторінці.

4. При завантаженні з youtube.com, користувач отримує відео у високій якості.

4. При завантаженні з файлообмінника, не потрібно чекати або інсталювати додаткові програми для завантаження.

5. Можливість інтеграції сервісу на особистий веб-сайт: користувачі сайту будуть отримувати прямі посилання для завантаження.

1.1.3. VSO Downloader Ultimate

Програма VSO Downloader Ultimate допоможе завантажити потокове відео з онлайн ресурсів у високій якості, конвертувати відео файли в будь-який формат і витягти аудіо доріжку (рис. 1.3).

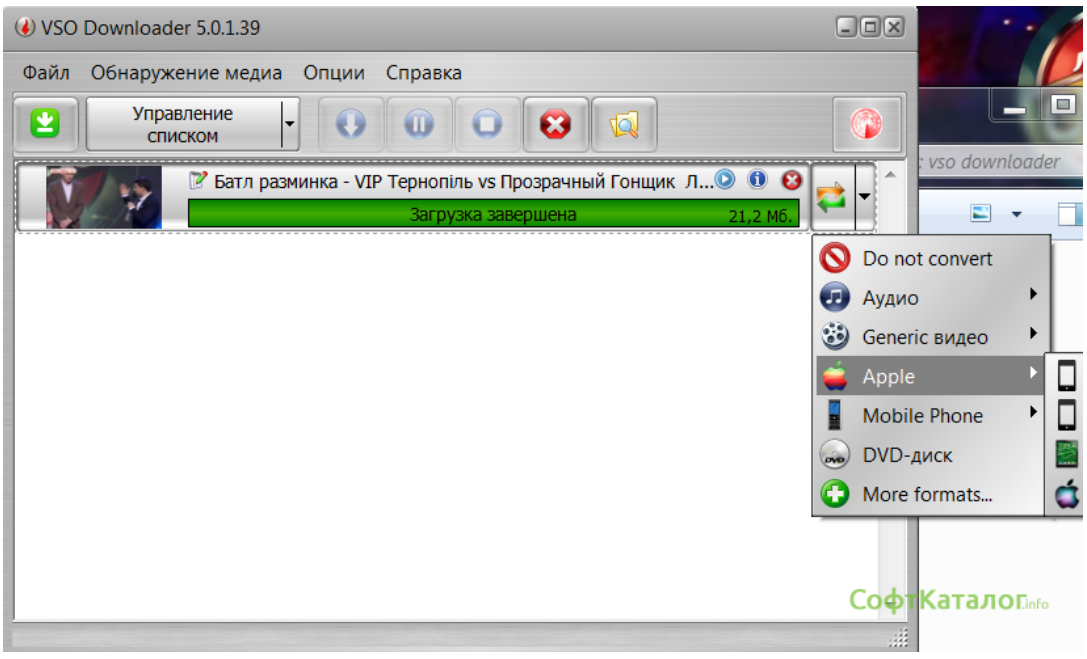


Рис. 1.3. VSO Downloader Ultimate

Можливості VSO Downloader Ultimate:

1. Відстеження посилань на завантаження в браузері (підтримує http https протоколи).
2. Поштучне і пакетне закачування роликів необмеженого розміру.
3. Управління списком-чергою файлів з можливістю постановки на паузу.
4. Перейменування файлів після завершення збереження, запис на DVD диск.
5. Конвертація форматів для програвання на мобільних пристроях.
6. Витяг аудіодоріжки в високій якості (бітрейт налаштовується).

Переваги:

1. Розумне автоматичне сканування браузера - копіювати і вставляти посилання не потрібно.
2. Багатопотоковий алгоритм скачування налаштований на оптимальне споживання системних ресурсів.
3. Інтегрований високошвидкісний движок завантаження, який працює в рази швидше аналогів.
4. Програма ігнорує банери і рекламні відео, але стоплист сайтів можна редагувати.
5. Процес завантаження файлів не вимагає втручання користувача, автоматичний режим добре працює.
6. Утиліта сумісна з Firefox, Chrome, Yandex Browser, Internet Explorer, Opera і багатьма іншими популярними оглядачами.
7. Завантажувач відео з youtube на ПК може зберігати відео у всіх популярних форматах.
8. Підтримується кілько тисяч сайтів для скачування мультимедійного контенту.
9. Можлива робота через VPN, проксі-сервер.

Недоліки:

1. Відсутність планувальника - VSO Downloader почне завантаження всього списку посилань тільки після підтвердження.
2. Прогалини в русифікації деяких пунктів меню.

1.1.4. Free Download Manager

Free Download Manager - безкоштовна програма, що поєднує в собі менеджер закачувань файлів і оффлайн браузер. Завдяки даній програмі ви зможете швидко і ефективно завантажувати як окремий файл або відразу декілька файлів з HTTP, HTTPS і FTP серверів, так і сайти цілком, а також завантажувати файли використовуючи BitTorrent протокол (рис. 1.4).

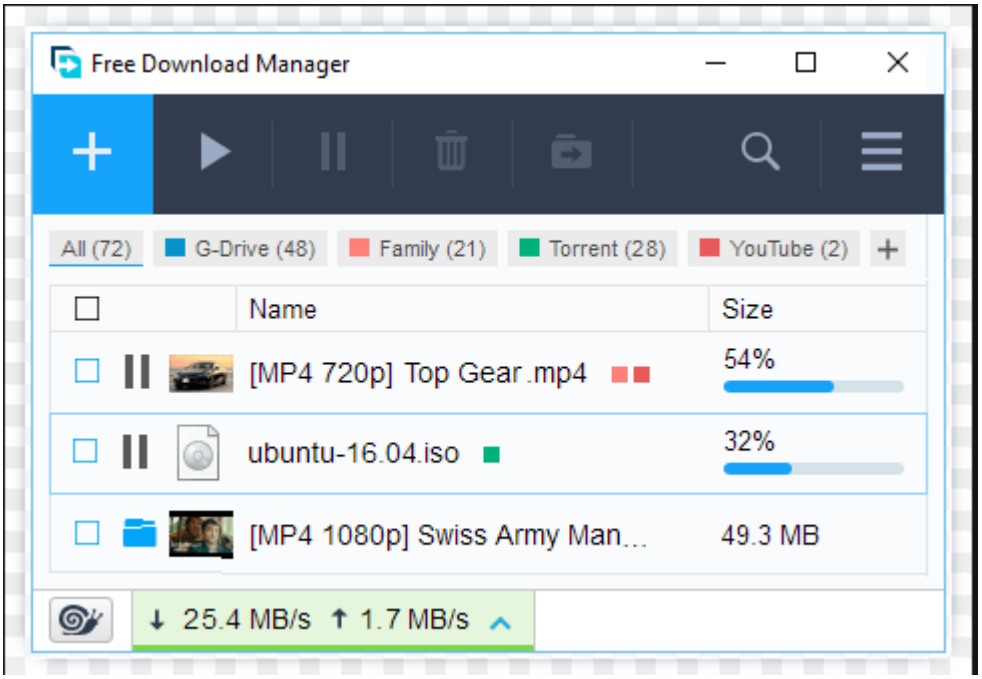


Рис. 1.4. Free Download Manager

Додаток підтримує роботу за розкладом, має багатий набір опцій для управління завантаженнями, підтримує відновлення перерваних закачувань і розбиття закачувань на потоки. При завантаженні мультимедійних файлів є можливість попереднього перегляду ще незавершених закачувань.

Крім цього, Free Download Manager дозволяє завантажувати відео з YouTube, Google Video, MySpace і багатьох інших подібних сервісів. При цьому, завантаження файл можна на льоту конвертувати в AVI з MPEG-4, AVI з XVID, FLV, WMV, MPEG1, MPEG2, MP4 для iPod / PSP пристроїв і т.д. з можливістю установки бажаного бітрейта і розміру відео.

Після інсталяції менеджер інтегрується в Windows Explorer, Outlook Express і браузер Internet Explorer, дозволяючи автоматично перехоплювати завантажувані файли або починати завантаження, викликаючи їх з контекстного меню браузера. Також присутня опція, аналогічна Кошику в FlashGet: для початку закачування файлу потрібно лише перетягнути посилання в невелике плаваюче віконце.

1.1.5. YTD Video Dowloader

YTD Video Downloader - програма для тих, кому необхідно завантажити відео з YouTube або інших сайтів (рис. 1.5).

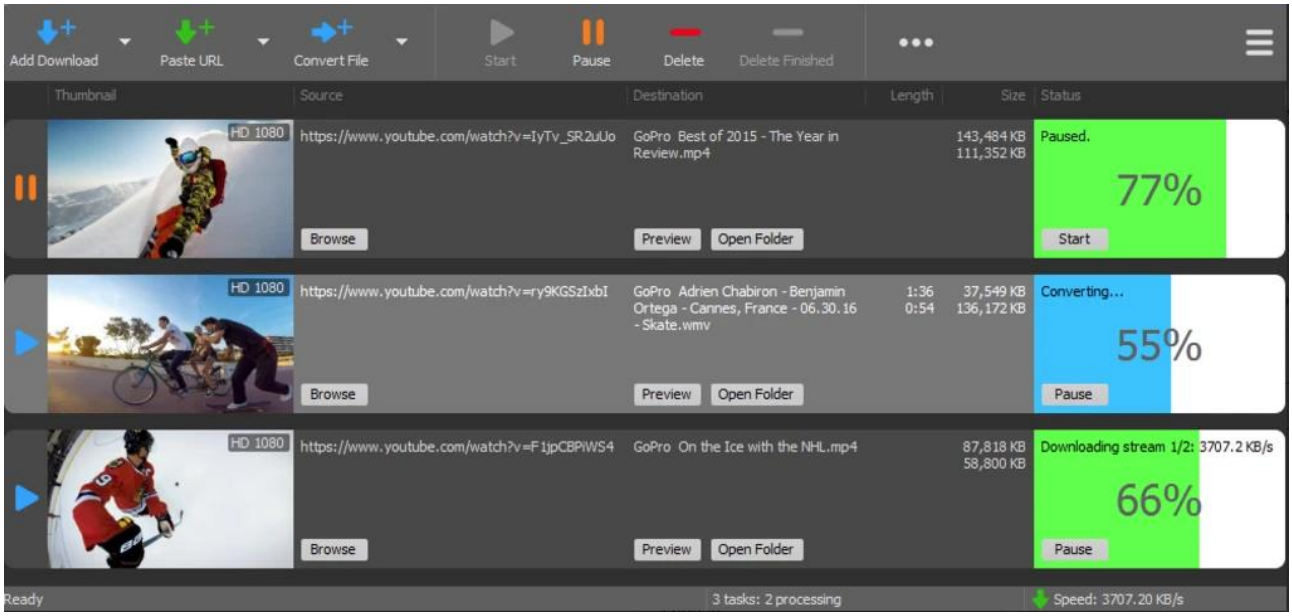


Рис. 1.5. YTD Video Dowloader

Особливістю програми є наявність вбудованого інструменту для конвертації завантажених відео. Це означає, що можна адаптувати відеоролики для відтворення на iPhone, iPad, PSP і інших пристроях, не використовуючи для цього додаткові програми.

Головні функції та особливості програми:

– Завантаження відео з більш, ніж 50 сайтів.

– Відображення прогресу скачування.

– Вибір якості відео.

– Конвертація відео в формати, сумісні з iPhone, PSP і іншими пристроями.

– Можливість відтворення викачаних відео.

– Витяг звукової доріжки з відео.

Різниця платної і безкоштовної версії полягає в кількості записів, які можна завантажити за один раз. У безкоштовній версії YTD кілька відео доведеться завантажувати по черзі. Розробник гарантує можливість завантаження більш ніж з 50 сайтів. Одна з найбільш зручних можливостей - витягти mp3 доріжку з відео.

1.2. Призначення розробки та область застосування

Статистичні дані показують масштаби перегляду відео в світі, а отже підтверджують потребу в повнофункціональній багатозадачній програмі для завантаження та перегляду онлайн відео без використання браузеру.

В розробці власного додатку використовується досвід програм, що були проаналізовані в п. 1.1.1. Співзначалися плюси та мінуси кожної з них, їхня популярність на ринку та можливості функціоналу. В результаті чого було зроблено висновок про необхідність створення програми для завантаження відео з різних ресурсів та перегляду онлайн без використання браузеру з використання мови програмування bash.

Дана програма призначена для завантаження й перегляду відео без використання браузеру та скачування з популярної мережі для перегляду відео Anilibria, Торрент трекерів, та інших ресурсів. Вона дозволяє змінювати озвучку та отримувати зображення хорошої якості, завдяки чому перегляд відео буде ще більш корисним та приємним, чого іноді важко досягнути при використанні браузеру.

Повноцінно розроблена програма Cool Video Downloader виконує функції менеджера завантажень та відеопрогравача і може стати в нагоді широкому спектру користувачів.

Дана робота має прикладний характер, оскільки її можна ефективно використовувати в повсякденному житті.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи бакалавра на тему «Розробка програмного забезпечення для завантаження та перегляду он-лайн відео без використання браузеру засобами Bash» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-_.

1.4. Постановка завдання

Завантаження відео через веб-браузер на високій швидкості і без збоїв з'єднання вдається не завжди і не на всіх сайтах. Тому найкращим вирішенням цієї проблеми є спеціальні програми для скачування відео з інтернету.

Аналіз ринку багатозадачних програми для скачування показав плюси та мінуси кожної з них та можливості функціоналу. В результаті чого було зроблено висновок про необхідність створення програми для завантаження відео з різних ресурсів та перегляду онлайн без використання браузеру з використання мови програмування bash.

Метою кваліфікаційної роботи є розробка програмного додатку для завантаження відео з різних ресурсів та його перегляду онлайн без використання браузеру.

Для досягнення мети було поставлено наступні завдання:

- дослідити особливості функціонування схожих програм;
- визначити найкращі способи оптимізації для програми;
- визначити специфіку роботи мови програмування Bash;
- змоделювати програмну платформу роботи;
- розробити програму;
- протестувати розроблену платформу для скачування та перегляду відео і порівняти її з іншими програмами шляхом визначення її переваг та недоліків.

В результаті виконання даної кваліфікаційної роботи повинен бути створений повнофункціональний багатозадачний програмний додаток Cool Video Downloader для завантаження й перегляду відео без використання браузеру.

Передбачається, що дана програма зможе відкривати доступ до перегляду в режимі онлайн та скачування мультемедійного контенту жанру аніме з популярного каталогу anilibria (https://www.anilibria.tv/).

Крім того, повинна існувати також можливість скачування відео з dark-libria та jutsu. Для цього необхідно передбачити детальний алгоритм виконання операцій та ходу програми.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Відповідно до поставленої мети потрібно розробити та реалізувати багатофункціональну програму для скачування і онлайн перегляду відео без відкриття браузеру.

Дана програма повинна забезпечувати наступний функціонал:

1. Виконувати функції менеджера завантажень.

2. Мати функції відеопрогравача.

3. Повинна підтримувати скачування з популярної мережі для перегляду відео Anilibria, Торрент трекерів, та інших ресурсів.

4. Дозволяти змінювати озвучення.

5. Отримувати зображення хорошої якості.

Головні функції та особливості програми:

– завантаження відео з каталогу anilibria, dark-libria та jutsu..

– відображення прогресу скачування.

– вибір якості відео.

– конвертація відео в формати, сумісні з iPhone, PSP і іншими пристроями.

– можливість відтворення викачаних відео.

– витяг звукової доріжки з відео.

Cool Video Downloader – розроблятиметься під операційну систему Linux та Windows.

Комунікація між користувачем та апаратною частиною програми буде проводитись в текстовому користувацькому інтерфейсі за допомогою уточнюючих команд.

1.5.2. Вимоги до інформаційної безпеки

Існує три основних вимоги для досягнення безпеки мережі:

1. Конфіденційність - тільки зазначені й авторизовані одержувачі можуть мати доступ до даних. Система безпеки має контроль доступу, а також має свої списки.

2. Цілісність - гарантія того, що інформація не була змінена в процесі передачі від вихідного пункту до місця призначення. Для цього використовується VPN та система IPs.

3. Доступність - своєчасний і надійний доступ до даних для

авторизованих користувачів. Це забезпечує списки контролю доступу та тумані обчислення, які роблять це швидко.

Антивірусна програма й анти-шпигунське ПЗ дозволяє запобігти зараженню кінцевих пристроїв шкідливими програмами.

1.5.3. Вимоги до складу та параметрів технічних засобів

Програма повинна використовуватися на ПК, що складається з наступних мінімальних технічних засобів та їх параметрів:

– CPU 1.6 + GHz;
– VGA 512 + Mb;
– RAM 512 + Mb;
– монітор з роздільною здатністю 1680/1050 і більше;
– клавіатура: стандартна - 101/102 клавіші або PS / 2 Microsoft Natural.
– миша.

1.5.4. Вимоги до інформаційної та програмної сумісності

Програмний додаток для завантаження та перегляду онлайн відео без використання браузеру Cool Video Downloader має бути виконаний засобами мови Bash та працювати під операційною системою Linux та Windows.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Метою кваліфікаційної роботи є розробка програмного додатку для завантаження відео з різних ресурсів та його перегляду онлайн без використання браузеру.

Дана програма забезпечує наступний функціонал:

1. Виконує функції менеджера завантажень.
2. Має функції відеопрогравача.
3. Підтримка скачування з популярної мережі для перегляду відео Anilibria, Торрент трекерів, та інших ресурсів.
4. Дозволяє змінювати озвучення.
5. Отримує зображення хорошої якості.

Особливості програми:

– завантаження відео з каталогу anilibria, dark-libria та jutsu..
– відображення прогресу скачування.
– вибір якості відео.
– конвертація відео в формати, сумісні з iPhone, PSP і іншими пристроями.
– можливість відтворення викачаних відео.
– витяг звукової доріжки з відео.

Комунікація між користувачем та апаратною частиною програми проводиться в текстовому користувацькому інтерфейсі за допомогою уточнюючих команд.

Програма Cool Video Downloader розроблена для використання на ОС Linux та Windows.

2.2. Опис застосованих математичних методів

В даному програмному додатку ні під час розробки, ні під час тестування та роботи додатка, не використовуються та не розглядаються ніякі математичні методи.

2.3. Опис використаної архітектури та шаблонів проектування

2.3.1. Етапи розробки програмного забезпечення

Розробка програмного забезпечення – важливий етап у створенні будь-якого програмного продукту. Важливо, щоб даний процес був ефективним, продуктивним та вигідним для компанії чи розробника. Саме тому всі етапи розробки програмного забезпечення було об’єднано під однією назвою – «життєвий цикл програмного забезпечення».

Життєвий цикл розробки програмного забезпечення - це застосування стандартної ділової практики для побудови програмних додатків. Зазвичай його поділяють на шість-вісім кроків: планування, вимоги (або ж цілі), дизайн, побудова (розробка продукту), поширення, випробування обслуговування. Деякі менеджери проектів можуть комбінувати, розділяти або опускати кроки, залежно від обсягу проекту. Це основні компоненти, рекомендовані для всіх проектів з розробки програмного забезпечення. [20]

ЖЦРПЗ - це спосіб вимірювання та вдосконалення процесу розробки. Він дозволяє провести дрібнозернистий аналіз кожного етапу процесу. Це, в свою чергу, допомагає компаніям та розробникам максимізувати ефективність на кожному етапі. Оскільки обчислювальна потужність зростає, це підвищує попит на програмне забезпечення та розробників. Компанії повинні скоротити витрати, швидше доставити програмне забезпечення та задовольнити або перевищити потреби своїх клієнтів. ЖЦРПЗ допомагає досягти цих цілей шляхом виявлення неефективності та вищих витрат та виправлення їх безперебійної роботи.

Життєвий цикл розробки програмного забезпечення просто окреслює кожне завдання, необхідне для складання програмного додатку. Це допомагає зменшити відходи та підвищити ефективність процесу розробки. Моніторинг також гарантує, що проект залишатиметься на шляху, і продовжує залишатися реальною інвестицією для компанії.

Багато розробників поділять ці кроки на менші підрозділи. Планування може бути розбито на дослідження технологій, маркетингові дослідження та аналіз витрат та вигод. Інші дії можуть зливатися між собою. Фаза тестування може працювати одночасно з фазою розробки, оскільки розробники повинні виправляти помилки, що виникають під час тестування [21]

Етап 1. Планування. На етапі планування керівники проекту оцінюють умови проекту. Це включає підрахунок трудових та матеріальних витрат, створення розкладу з цільовими завданнями та створення команд проекту та структури керівництва. Планування може також включати відгуки зацікавлених сторін. Зацікавлені сторони - це всі, хто може отримати вигоду від програми. Планування має чітко визначати обсяг і мету проекту. Він складає план курсу та положення команди для ефективного створення програмного забезпечення. Він також встановлює межі, які допомагають запобігти розширенню або зміщенню проекту від початкової мети.

Етап 2. Визначення вимог. Визначення вимог вважається частиною планування для визначення того, що повинна робити програма, та її вимог. Наприклад, програма в соціальних мережах потребує можливості зв’язку між різними користувачами. Інвентаризаційна програма може вимагати функції пошуку тощо. Вимоги також включають визначення ресурсів, необхідних для побудови проекту. Наприклад, команда може розробити програмне забезпечення для управління спеціальною виробничою машиною. Машина є вимогою в процесі.

Етап 3. Дизайн та прототипування. Етап проектування моделює спосіб роботи програмного додатку. Деякі аспекти дизайну включають:

- Архітектуру - визначає мову програмування, галузеві практики, загальний дизайн та використання будь-яких шаблонів чи нарисів.
- Інтерфейс користувача - визначає способи взаємодії клієнтів із програмним забезпеченням та реакцію програмного забезпечення на подразники.
- Платформи - визначає платформи, на яких буде працювати програмне забезпечення, такі як Apple, Android, версія Windows, Linux або навіть ігрові консолі.
- Програмування - не просто мова програмування, але включаючи методи вирішення проблем та виконання завдань у додатку.
- Зв'язок - визначає методи, за допомогою яких програма може взаємодіяти з іншими активами, такими як центральний сервер або інші екземпляри програми
- Безпека - визначає заходи, вжиті для захисту програми, і може включати шифрування трафіку SSL, захист паролем та безпечне зберігання облікових даних користувача

Прототипування може бути частиною етапу проектування. Прототип схожий на одну з ранніх версій програмного забезпечення в моделі розробки програмного забезпечення Iterative. Він демонструє основне уявлення про те, як виглядає та працює програма. Цей "практичний" дизайн можна показати зацікавленим сторонам.

Етап 4. Розробка програмного забезпечення. Це власне написання програми. Невеликий проект може бути написаний одним розробником, тоді як великий проект може бути розбитий та опрацьований кількома командами. На цьому етапі використовують програму контролю доступу або управління вихідним кодом. Ці системи допомагають розробникам відстежувати зміни коду. Вони також допомагають забезпечити сумісність між різними командними проектами та забезпечити досягнення цільових завдань.

Процес кодування включає багато інших завдань. Багатьом розробникам потрібно попрацювати над навичками або працювати в команді. Пошук та

виправлення помилок та збоїв є критичним. Завдання часто затримують процес розробки, наприклад, очікування результатів тестування або складання коду для запуску програми. ЖЦРПЗ може передбачити ці затримки, щоб розробники могли отримати інші завдання.

Розробники програмного забезпечення цінують інструкції та пояснення. Документація може бути формальним процесом, включаючи підключення посібника користувача для програми. Вона також може здійснюватися неформально, як коментарі у вихідному коді, що пояснюють, чому розробник використовував певну процедуру. Навіть компанії, які прагнуть створити легке та інтуїтивно зрозуміле програмне забезпечення, користуються документацією.

Документація може бути швидким оглядом основних функцій програми, які відображаються при першому запуску. Це можуть бути відеоуроки для складних завдань. Письмова документація, така як посібники користувача, посібники з усунення несправностей та поширені запитання, які допоможуть користувачам вирішити проблеми або технічні питання.

Етап 5. Тестування. Дуже важливо протестувати програму, перш ніж зробити її доступною для користувачів. Значну частину тестування можна автоматизувати, як тестування безпеки. Інші випробування можна проводити лише в певному середовищі - розглядають можливість створення імітованого виробничого середовища для складних розгортань. Тестування повинно забезпечити правильну роботу кожної функції. Також слід протестувати різні частини програми для безперебійної роботи разом - перевірки продуктивності, щоб зменшити будь-які зависання або затримки в обробці. Етап тестування допомагає зменшити кількість помилок та збоїв, з якими стикаються користувачі. Це призводить до вищого задоволення користувачів та кращого рівня використання.

Етап 6. Поширення. На етапі поширення додаток стає доступним для користувачів. Багато компаній воліють автоматизувати фазу поширення. Вона може виглядати, як платіжний портал та посилання для завантаження на веб-сайті компанії. Або завантаження програми на смартфон. Поширення також

може бути складним. Одним із прикладів є оновлення бази даних на рівні компанії до нещодавно розробленої програми. Оскільки в базі даних використовується кілька інших систем, інтеграція оновлення може зайняти більше часу та зусиль.

Етап 7. Експлуатація та обслуговування. На цьому цикл розробки майже закінчений. Додаток зроблено і використовується в полі. Однак фаза експлуатації та технічного обслуговування все ще важлива. На цьому етапі користувачі виявляють помилки, які не були знайдені під час тестування. Ці помилки потрібно вирішити, що може породити нові цикли розробки. [20]

2.3.2. Опис архітектури додатку

Для взаємодії між собою сучасні комп'ютери використовують комп'ютерні (обчислювальні) мережі. Основним призначенням таких мереж є:

1. Надання доступу до інформації.
2. Сумісне використання технічних ресурсів.
3. Розподілене навантаження.
4. Віддалене керування.
5. Забезпечення надійності.

Основними можливостями є:

1. Можливість швидкої передачі інформації на великі відстані.
2. Оперативний пошук інформації.
3. Обмін текстовою, звуковою та відеоінформацією у реальному часі.
4. Зберігання інформації.

Мережі класифікують за типом використаних протоколів, топологією, областю дії, швидкістю передачі даних, призначенням, тощо. За архітектурою вони бувають:

1. Однорангові.
2. Клієнт-серверні.

Однорангові системи – системи які складаються з рівноправних вузлів. В

ній всі учасники є однаковими, на відміну від клієнт-серверної архітектури, де лише окрема категорія учасників (сервера) можуть передавати інформацію іншим.

Клієнт-серверна архітектура є найпопулярнішою серед всіх архітектур. Вона передбачає наступні компоненти:

1. Набір серверів, які оброблюють та надають інформацію.
2. Набір клієнтів, які використовують надану сервером інформацію.
3. Мережа в якій передається інформація.

В більшості сучасних програм є дуже важливим обмін інформацією в реальному часі – так звані «real-time» системи. Нажаль в відкритих джерелах не існує більш-менш повної інформації про проектування архітектури real-time систем. Є багато статей на окремі теми, але дуже важко знайти хоча б одну, що їх об’єднує. Кожному розробнику потрібно по частинкам збирати цю інформацію та проектувати систему самостійно, а великі компанії не прагнуть відкривати світові секрети архітектури своїх продуктів. Тим не менш, обмін інформацією між сервером та клієнтом є невід’ємною частиною великої кількості сучасного програмного забезпечення.

2.3.3. Забезпечення клієнт-серверного з’єднання

Для завантажування дані програми використовують більш оптимізовані та стиснуті файли, які містять інформацію, пов’язану з іншими файлами та папками, що підлягають розповсюдженню.

Коли користувач використовує файл, це майже означає, що він передає файл через мережу P2P або BitTorrent.

Peer-to-peer, P2P - варіант архітектури системи, в основі якої стоїть мережа рівноправних вузлів (рис. 2.1).

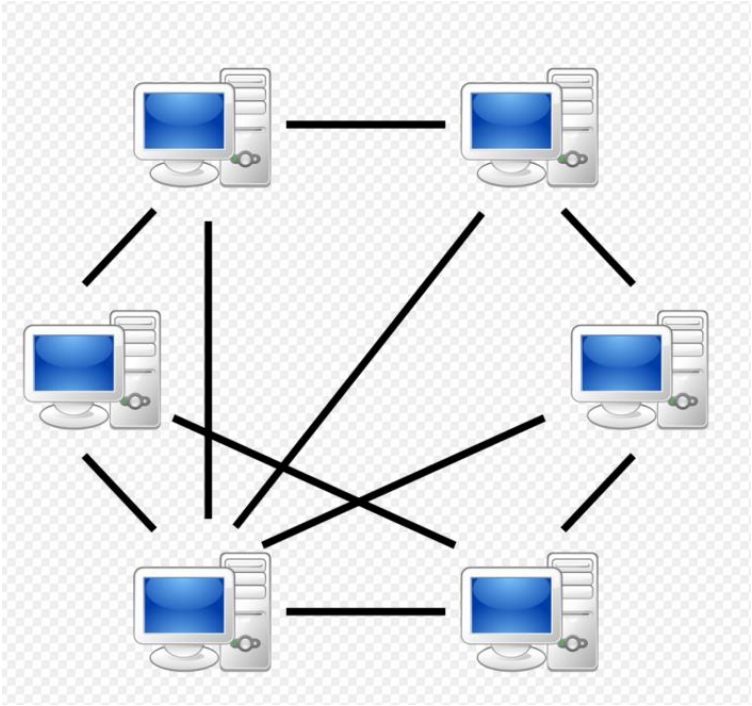


Рис. 2.1. Архітектура мережі по типу P2P

Комп'ютерні мережі типу peer-to-peer (або P2P) засновані на принципі рівноправності учасників і характеризуються тим, що їх елементи можуть зв'язуватися між собою, на відміну від традиційної архітектури, коли лише окрема категорія учасників, яка називається серверами, може надавати певні сервіси іншим.

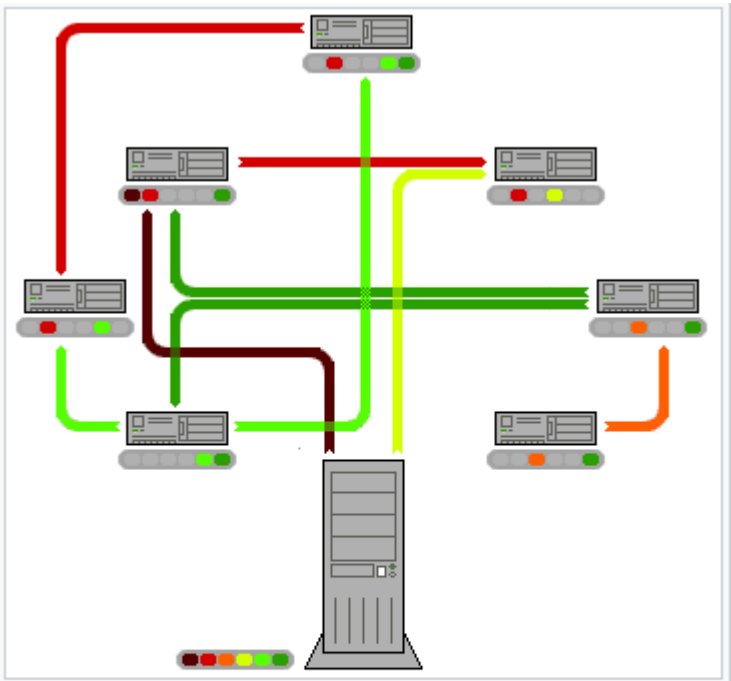
В чистій мережі «peer-to-peer» не існує поняття клієнтів або серверів, лише рівні вузли, які одночасно функціонують як клієнти та сервери по відношенню до інших вузлів мережі. Ця модель мережевої взаємодії відрізняється від клієнт-серверної архітектури, в якій зв'язок відбувається лише між клієнтами та центральним сервером. Така організація дозволяє зберігати працездатність мережі при будь-якій конфігурації доступних її учасників. Проте практикується використання P2P-мереж, які все ж таки мають сервери, але їх роль полягає вже не у наданні сервісів, а у підтримці інформації з приводу сервісів, що надаються клієнтами мережі.

В системі P2P автономні вузли взаємодіють з іншими автономними вузлами. Вузли є автономними в тому сенсі, що не існує загальної влади, яка

може контролювати їх. В результаті автономії вузлів вони не можуть довіряти один одному та покладатися на поведінку інших вузлів, тому проблеми масштабування та надмірності стають важливішими ніж у випадку традиційної архітектури.

Сучасні P2P-мережі набули розвитку завдяки ідеям, пов'язаними з обміном інформацією, які формувалися у руслі того, що кожен вузол може надавати й отримувати ресурси, які надаються будь-якими іншими учасниками.

BitTorrent - відкритий протокол обміну інформацією у мережах типу peer-to-peer (рис. 2.2).



Анімація використання протоколу: кольорові точки під кожним комп'ютером в анімації відповідають різним частинам файлу, який роздається. Трекер (сервер) надає тільки одну копію файлу, і всі користувачі клонують його частини один від одного. Показано, наскільки швидше файл буде завантажено, якщо використовується протокол BitTorrent.

Рис. 2.2. Процес використання протоколу BitTorrent

Протокол розробляли таким чином, аби обмін файлами великих розмірів у мережі був полегшений для її учасників. Один із принципів роботи протоколу BitTorrent такий: навантаження на учасника, який розповсюджує певний файл,

зменшується завдяки тому, що клієнти, які його скачують, починають обмінюватися даними між собою одразу, навіть поки файл повністю не скачано. Таким чином, клієнти, які скачали певну частину великого файлу, одразу можуть бути джерелами його розповсюдження.

Така ідея організації протоколу має переваги порівняно з протоколами peer-to-peer мереж першого покоління, де файл скачується з одного розповсюджувача чи з декількох розповсюджувачів частинами.

Для отримання інформації про розповсюджувачів певного файлу, клієнт може звернутися до так званих трекерів.

Трекер (англ. tracker) - спеціалізований сервер, який працює по HTTP протоколу. Трекер використовується для того, щоб клієнти могли знайти один одного. На трекері зберігаються IP-адреси клієнтів, вхідні порти клієнтів та хеш-суми, які унікальним чином ідентифікують об'єкти, що беруть участь у скачуваннях. За стандартом, імена файлів на трекері не зберігаються, та дізнатися їх по хеш-сумам не можна. Проте на практиці часто трекер, окрім своєї основної функції, виконує також функцію невеличкого веб-серверу. Такий сервер зберігає файли метаданих, що містять значення хеш-функції, та разом з ними опис файлів, які розповсюджуються, кількість розповсюджувачів, статистику завантажень тощо.

Перед початком завантаження файлу клієнт з'єднується з трекером, повідомляє йому свою IP-адресу та хеш-суму файла, що завантажується. У відповідь клієнт отримує адреси інших учасників мережі, які розповсюджують або закачують той самий файл. Далі клієнт періодично інформує трекер про хід процесу завантаження та отримує оновлений перелік адрес.

Клієнти з'єднуються один з одним та обмін даними відбувається без безпосередньої участі трекера. Учасники закачування обмінюються інформацією про наявність сегментів файлу. Клієнт, який бажає закачати певний фрагмент, надсилає запит, і, якщо інший клієнт готовий його надати, відбувається процес закачування. Після цього клієнт перевіряє контрольну суму

сегменту та сповіщає всіх приєднаних учасників закачування про його наявність.

Для ефективної роботи мережі BitTorrent необхідно, щоб якомога більше клієнтів були здатні приймати вхідні з'єднання. Неправильна настройка NAT чи файрволу можуть цьому заважати.

Алгоритм обміну даними:

1. Кожен клієнт має можливість тимчасово блокувати віддачу іншому клієнтові (англ. choke). Це робиться для ефективнішого використання каналу віддачі. Крім того, при виборі — кого розблокувати, перевага віддається пірам, які самі передали цьому клієнтові багато сегментів. Таким чином, піри з хорошими швидкостями віддачі заохочують один одного за принципом «ти - мені, я - тобі».

2. Обмін сегментами ведеться за принципом «ти - мені, я - тобі» симетрично в двох напрямках. Клієнти повідомляють один одному про наявні у них сегменти при підключенні та потім при отриманні нових сегментів, і тому кожен клієнт може зберігати інформацію про те, які сегменти є у інших підключених пірів. Порядок обміну обирається таким чином, щоб спочатку клієнти обмінювалися найрідкіснішими сегментами: таким чином підвищується доступність файлів в роздачі . Водночас вибір сегмента серед найрідкісніших випадковий, і тому можна уникнути ситуації, коли всі клієнти починають завантажувати один і той же самий рідкісний сегмент, що мало б негативний вплив на продуктивність.

Обмін даними починається, коли обидві сторони в ньому зацікавлені, тобто, кожна зі сторін має сегменти, яких немає в іншої. Кількість переданих сегментів підраховується, і якщо одна із сторін виявляє, що передає в середньому більше, ніж приймає, вона блокує на деякий час віддачу іншій стороні. Таким чином, в протокол закладено захист від лічерів.

Сегменти діляться на блоки розміром 16-4096 кілобайт, і кожен клієнт запитує саме ці блоки. Одночасно можуть запитуватися блоки з різних сегментів. Більш того, деякі клієнти підтримують скачування блоків одного

сегмента у різних пирів. У цьому випадку описані вище алгоритми і механізми обміну застосовуються і до рівня блоків.

2.4. Опис використаних технологій та мов програмування

2.4.1. Методи оптимізації програмного забезпечення

З перших днів розвитку ери обчислювальної техніки виникло питання економії місця і збільшення продуктивності програм. Програмістам доводилося створювати складні дієздатні програми, які змогли б працювати при дуже низькому швидкодії процесорів, використовувати лічені кілобайти оперативної пам'яті і місця на диску. Тому всі розробники ПЗ були зацікавлені в максимальній швидкодії при мінімальному розмірі коду.

Сьогодні ці потужності викликають посмішку. Але традиції оптимізації коду збереглися. Як відомо, скільки не нарощуй розмір диска і обсяг оперативної пам’яті, все одно буде мало. Тому написані «неохайні» додатки, повільні і ресурсомісткі, програють конкурентну боротьбу аналогам, навіть якщо вони красиві і зручні. Особливо жорсткі вимоги стосуються драйверів і системних утиліт. Вони повинні працювати швидко, коректно і максимально економити ресурси комп'ютера. Тобто взаємодія процесора з периферією має відбуватися без зайвих витрат часу, з високою швидкістю передачі даних між пристроями.

Основні принципи оптимізації - природність, продуктивність, витрачений час. Природність означає, що код повинен бути акуратним, модульним і легко читабельним. Кожен модуль повинен природно вбудовуватися в програму. Код повинен легко піддаватися редагуванню, інтегруванню або видаленню окремих функцій та можливості без необхідності вносити серйозні зміни в інші частини програми.

Продуктивність означає, що результаті оптимізації необхідно отримати приріст продуктивності програмного продукту. Як правило, вдало оптимізована програма збільшує швидкодію мінімум на 20-30% в порівняння з вихідним

варіантом. Час означає, що оптимізація і подальші налагодження повинні займати невеликий період часу. Оптимальними вважаються терміни, що не перевищують 10 - 15% часу, витраченого на написання самого програмного продукту. Інакше це буде не рентабельно.

Отже, існує велика кількість методів оптимізації. Серед них:

1. Налаштування оточення

Налаштування за замовчуванням далеко не ідеальні. Можливо, якісь додаткові перевірки якраз і уповільнюють процес. Іноді вдається помітно прискорити роботу програми, змінивши ключові настройки. До речі, прискорює роботу не тільки тестованої програми, але і всієї системи реалізації.

2. Мемоізація

Мемоізація (від англ. Memoization) означає запам'ятовування. Фактично це просте збереження результату виконання певної функції, яке допоможе уникнути її повторного виконання. Застосовуючи мемоізацію, можна значно підвищити продуктивність програми.

Працює це дуже просто. Перед тим, як функція буде виконуватися, перевіряється умова - чи виконувалася вона раніше. За підсумками можна отримати два варіанти:

– функція викликана в перший раз, тоді вона виконується, а результат зберігається;

– модуль вже працював, можна використовувати збережений результат.

Іноді говорять про табулювання, це синонім мемоізації, який використовується в багатьох мовах програмування.

3. Кешування

Це метод тимчасового зберігання даних в пам'яті пристрою користувача. Отримати доступ до такої інформації можна набагато швидше, ніж кожен раз звертатися до сервера або баз даних. За допомогою кешування значно прискорюють роботу з сайтами, онлайн-системами і т.д.

Вся необхідна інформація в даному випадку зберігається на носіях з швидким доступом. Це може бути виділена частина диска або оперативна пам'ять. Програма в процесі роботи використовує кеш в міру необхідності, і звертається до основного сховища даних тільки якщо не знаходить їх в кеші.

4. Розпаралелювання програм

Це спосіб адаптації алгоритмів, які були реалізовані, як програми для комп'ютерних систем з паралельною архітектурою. Як правило, це стосується багатопроцесорних систем.

5. Метод наближення

Наближення або апроксимація (від лат. Proxima - найближча або наближення) - метод заміни суворого алгоритму на найбільш підходящі наближені значення, що тягне за собою певну втрату точності обчислень. Зниження точності економить пам'ять і підвищує швидкість. Для того щоб не використовувати довгу арифметику, можна скористатися звичайними float'ами.

6. Використання сторонніх мов

Іноді написана програма може повільно працювати через те, що багато часу займає перевірка описаних типів, що займає додатковий час. Щоб уникнути цього ефекту, можна застосовувати фрагменти коду або модулі, написані на інших мовах. Але робити це потрібно вкрай обережно. Всі ці «зайві» перевірки захищають вас від багів і «дірок» в безпеці, пов'язаних, в тому числі, з буферизацією. Крім того, якщо почати використовувати в коді фрагменти інших мов, це може викликати ефект «зоопарку», що сильно знижує читабельність програми.

2.4.2. Опис використаної мови програмування

Для розробки програми Cool Video Downloader була вибрана мова програмування Bash.

Bash - це оболонка Unix та мова команд, написана Брайаном Фоксом для проекту GNU як безкоштовна програма для заміни оболонки Bourne. Вперше

випущений в 1989 році, він використовувався як оболонка входу за замовчуванням для більшості дистрибутивів Linux та всіх версій macOS від Apple до macOS Catalina. Також доступна версія для Windows 10 через підсистему Windows для Linux. Це також оболонка користувача за замовчуванням у Solaris 11. [14]

Bash - це командний процесор, який зазвичай працює у текстовому вікні, де користувач вводить команди, що викликають дії. Bash також може читати та виконувати команди з файлу, який називається сценарієм оболонки. Як і всі оболонки Unix, він підтримує глобалізацію імені файлу (збіг із підстановкою), конвеєр, тут документи, заміну команд, змінні та структури управління для перевірки стану та ітерації. Ключові слова, синтаксис, динамічно змінні змінні та інші основні особливості мови копіюються з sh. Інші функції, наприклад, історія, копіюються з csh та ksh. Bash - це оболонка, сумісна з POSIX, але з низкою розширень.

Ім'я оболонки - це скорочення від Bourne Again Shell, каламбур назви оболонки Борна, яку вона замінює , і поняття "народитися знову". [11]

Командна оболонка Bash (/ bin / bash) розроблена для того, щоб подолати обмеження попередніх оболонок. Продовжуючи традиції Bourne shell, Bash надає наступні переваги:

1. Потужний і простий у використанні мова сценаріїв.
2. Переваги інтерактивної взаємодії, запозичені з сімейства командних оболонок C Shell.
3. Свобода поширення і повністю відкриті вихідні коди.
4. Повна реалізація специфікації POSIX 1003.2.

Завдяки багатій історії розвитку Bash надає ряд корисних особливостей, які покращують взаємодію користувача з командною оболонкою та спрощують розробку сценаріїв - простих програм, що автоматизують призначені для користувача завдання.

Bash легко налаштовується - як через спеціальні файли ініціалізації, так і через опції настройки.

Робота в командному рядку Bash: Bash - це програма, яка працює в терміналі UNIX. Отже, щоб почати роботу з Bash, необхідно запустити новий термінал. Система спочатку може бути налаштована на роботу за замовчуванням з іншого командною оболонкою. Щоб дізнатися, яка командна оболонка в дійсності запущена, потрібно набрати echo $ SHELL в командному рядку терміналу і потім натиснути клавішу Enter. Якщо у відповідь буде виведено / bin / bash, то запущеної командною оболонкою є Bash. Якщо отриманий інший відповідь від системи, значить, використовується інша командна оболонка.

Щоб дізнатися, чи встановлена Bash в системі, можна ввести whereis bash в командному рядку. Ця команда знаходить Bash і виводить повний шлях до програми. Якщо Bash не встановлена в системі, можна вільно завантажити і встановити її новітню версію. Якщо Bash встановлена в системі, можна змінити запущену за замовчуванням командну оболонку на Bash, ввівши команду chsh bash. Необхідно відзначити, що в деяких системах, таких як MAC OS X, написання команди може бути трохи іншим. Запустити командну оболонку Bash можна також (якщо Bash встановлена в системі, навіть якщо вже запущена інша командна оболонка), ввівши bash в командному рядку і натиснувши Enter.

Коли Bash запущена, дуже просто налаштувати різні опції її використання. Щоб побачити список опцій настройок, необхідно ввести команду set -o в командному рядку Bash.

Синтаксис команди Bash є надмножиною синтаксису команди Bourne. Bash підтримує розширення фігурних дужок, заповнення командного рядка (Програмоване завершення), основну налагодження та обробку сигналів (за допомогою пастки), серед інших функцій bash 2.05a . Bash може виконувати переважну більшість сценаріїв оболонки Bourne без змін, за винятком сценаріїв оболонки Bourne, які натрапляють на поведінку синтаксису бахроми, що інтерпретується в Bash по-різному, або намагається запустити системну команду, що відповідає новій вбудованій Bash, тощо. з оболонки KornShell (ksh) та оболонки C (csh), таких як редагування командного рядка, історія

команд (команда історії), стек каталогів, змінні $ RANDOM і $ PPID, а також синтаксис заміни команд POSIX $ (…) .

Коли користувач натискає клавішу табуляції в інтерактивній командній оболонці, Bash автоматично використовує завершення командного рядка, починаючи з бета-версії 2.04, щоб відповідати частково набраним назвам програм, іменам файлів та іменам змінних. Система завершення командного рядка Bash є дуже гнучкою та настроюваною, і вона часто комплектується функціями, які заповнюють аргументи та імена файлів для конкретних програм та завдань.[15]

Синтаксис Bash має безліч розширень, яких не вистачає в оболонці Борна. Bash може виконувати цілочисельні обчислення ("арифметична оцінка"), не створюючи зовнішніх процесів. Для цього використовується команда ((...)) та синтаксис змінної $ ((...)). Його синтаксис спрощує переспрямування вводу-виводу. Наприклад, він може одночасно перенаправляти стандартний вивід (stdout) і стандартну помилку (stderr) за допомогою оператора &>. Це простіше набрати, ніж еквівалент оболонки Борна 'command> file 2> & 1'. Bash підтримує заміну процесу, використовуючи синтаксис <(команда) та> (команда), який замінює вихід (або введення в) команди, де зазвичай використовується ім'я файлу. (Це реалізується за допомогою/proc/fd / неназваних каналів у системах, що підтримують це, або за допомогою тимчасових іменованих каналів, де це необхідно)

При використанні ключового слова 'function' оголошення оголошень Bash не сумісні зі сценаріями Bourne/Korn/POSIX (у KornShell виникає та сама проблема при використанні 'function'), але Bash приймає той самий синтаксис оголошення функції, що і оболонки Bourne / Korn, і відповідає POSIX. Через ці та інші відмінності, сценаріями оболонки Bash рідко можна керувати під інтерпретаторами оболонки Bourne або Korn, якщо це не було навмисно написано з урахуванням цієї сумісності, яка стає все рідше, оскільки Linux стає все більш поширеним. Але в режимі POSIX Bash тісніше відповідає POSIX.[7]

Bash підтримує in документи. Починаючи з версії 2.05b Bash може перенаправляти стандартне введення (stdin) з " in рядка" за допомогою оператора <<<.

Bash 3.0 підтримує відповідність регулярних виразів у процесі, використовуючи синтаксис, що нагадує Perl.

У лютому 2009 року Bash 4.0 представив підтримку асоціативних масивів. Індекси асоціативних масивів - це рядки, подібні до AWK або Tcl. Вони можуть використовуватися для емуляції багатовимірних масивів. Bash 4 також перемикає свою ліцензію на GPLv3; деякі користувачі підозрюють, що ця зміна ліцензії є тим, чому MacOS продовжує використовувати старіші версії.

2.4.3. Опис використаних технології для розробки ПЗ

2.4.3.1. Технології скачування та зберігання плейлистів

Для можливості скачування та перегляду відео в режимі онлайн багатозадачному скрипту програми Cool Video Downloader потрібно завантажити m3u8 потік.

M3U - формат комп'ютерного файлу для зберігання плейлистів. Спочатку формат з'явився в мультимедійних програвачів Winamp, але з часом його стали підтримувати майже всі інші програвачі.

Файл M3U - це звичайний текстовий файл в кодуванні Latin-1, що містить шлях до одного або більше мультимедійних файлів, які належить відтворити програвачу. Кожен шлях розташований на окремому рядку. Шляхи можуть бути абсолютними, відносними (наприклад, «C:\Музика\ песня.mp3» або «песня.mp3»), а також URL-адресами. Файл може містити коментарі, що починаються з символу «#». У extended M3U символ «#» також означає внутрішні директиви.

Один з найпоширеніших способів використання формату M3U - створення плейлистів, що містять один-єдиний запис, що веде на потокове мультимедіа-мовлення в Інтернеті. Такий файл надає можливість легко

обмінятися посиланням на потокове мовлення по електронній пошті або через файлообмінні мережі.

Файл має розширення «M3U» або «m3u». При редагуванні m3u-файлу в текстових редакторах, наприклад в Блокноті, слід зберігати результат в кодуванні ANSI (Windows -1252), сумісної для звичайних текстів без спеціальних символів з кодуванням Latin-1. Файли в Unicode-модифікації формату, що використовує кодування UTF-8 замість кодування Latin-1, мають розширення «m3u8».

2.4.3.2. Забезпечення маніпуляцій з цифровими відео та аудіо даними

Для успішного завантаження відео, користуючись даною бібліотекою необхідно проводити згодом ряд конвертацій. Для цього в програмі Cool Video Downloader використовується бібліотека ffmpeg.

FFmpeg - це комплекс вільних комп'ютерних програм та програмних бібліотек для маніпуляцій з цифровими відео- та аудіо-матеріалами - запис, конвертація та пакування у різні формати контейнерів.

Проєкт славиться наявністю різних аудіо та відео кодеків.

Інтерфейс командного рядка має інтуїтивний вигляд.

FFmpeg було розроблено під Linux, але він успішно працює й у Apple Mac OS X та Microsoft Windows.

Цей проєкт складається із декількох компонент:

1. ffmpeg - програма командного рядка для конвертування одного формату відео у інший. Вона також дозволяє захоплювати і кодувати відео в режимі реального часу від декількох апаратних і програмних джерел, таких як карта захоплення ТБ.

2. ffserver - мультимедійний сервер трансляції HTTP і RTSP, що дозволяє здійснювати живі чи записані трансляції. Він також може використовуватись для зсуву у часі прямої трансляції.

3. ffplay - простий медіа програвач, який використовує в собі SDL і бібліотеки FFmpeg.

4. ffprobe - інструмент командного рядка для зображення медіа інформації (тексту, CSV, XML, JSON), див. також Mediainfo.

2.4.3.3. Застосування парсингу контенту

Парсинг (від англ. Parse) – процес аналізу або розбору певного контенту на складові за допомогою роботів-парсерів (спеціальних програм або скриптів). У SEO цим контентом є html-код сторінок сайтів. Найвідоміші парсери в мережі – це пошукові роботи, які аналізують сторінки, зберігають дані аналізу у себе в базі і потім при пошуку видають релевантні та актуальні документи. Часто парсинг плутають з граббінгом. Це близькі поняття, але все ж мають різні значення. Граббер дозволяє скачувати інформацію з мережі (html-сторінки, rss-стрічки, xml-документи) в свою базу, а парсер дозволяє виявити з цієї купи корисну інформацію і обробити її, залежно від поставлених завдань. У галузі пошукової оптимізації парсинг використовується дуже часто. Всі SEO-інструменти щось парсингують (посилання, ключові слова) і на основі цього надають корисні дані для аналізу.

Парсер це скрипт, призначення якого полягає в автоматичному створенні статей на сайті. Парсер, за заданими параметрами, виконує пошук в мережі Інтернет потрібного контенту та переносить його на вказаний сайт, тому парсер часто називають грабером сайтів. Застосування скриптів-парсерів дозволяє швидко наповнити новий сайт великою кількістю тематичної інформації практично без участі веб-розробника.

В результаті використанні парсерів веб-розробник отримує сайт, вся інформація якого вже існує в мережі. Такий метод отримання контенту не схвалюється пошуковими системами, так як отримані тексти не є унікальними. Ранжування пошуковими системами сайтів, вся інформація яких отримана з допомогою парсерів, завжди буде дуже низькою. За плагіат інформації такий

сайт навіть може бути вилученим з результатів пошуку.

У більшість сучасних парсерів є функції автоматичної обробки отриманих текстів перед їх викладенням на сайт. Для цього вони застосовують різні алгоритми автоматизованого рерайтингу статей. Однак, яким би гарним не був алгоритм рерайтингу, він завжди поступається контенту написаному людиною.

Використання парсеру може бути виправданим для підтримки актуальності сайту. Копіювання свіжих новин зі сторонніх ресурсів може призвести до ситуації коли вони будуть першими проіндексовані на вашому сайті, а не на сайті з якого отримані. В результаті чого пошукова система буде вважати оригіналом контенту викрадену статтю.

Крім того, постійна публікація свіжих актуальних новин також позитивно сприймається пошуковими системами.

Застосування скриптів-парсерів може бути виправданим для наповнення ресурсів, для яких просування в пошукових системах не є актуальним. Наприклад, форумів або блогів соціальних спільнот, де користувачі можуть отримати актуальну інформацію агреговану з різних інтернет-сайтів.

2.4.3.4. Використання утиліт для організації вибірки даних з мережі без браузеру

cURL - назва проекту і крос-платформового програмного засобу, що служить для передачі даних через Інтернет. cURL - це утиліта для організації вибірки даних з вебу, що надає можливість гнучкого формування запиту із завданням таких параметрів, як cookie, user_agent, referrer і будь-яких інших заголовків. cURL - це додаткова можливість оперувати з файлами на стороні сервера сторінок Інтернету за допомогою параметрів, що можуть бути переданими в рядку URL. За допомогою cURL можна, наприклад, отримати html-сторінку, не використовуючи для цього браузер.

grep - утиліта інтерфейсу командного рядка, яка знаходить на вводі рядки, що відповідають заданому регулярному виразу, і виводить їх. Назва утиліти є послідовністю команд пошуку регулярних виразів у редакторі ed - g/re/p. Цю послідовність команд можна описати англійською фразою «search globally for lines matching the regular expression, and print them» - «шукати скрізь рядків, відповідних регулярному виразу, і виводити їх».

Спочатку була створена для операційної системи UNIX.

Існують модифікації grep:

– egrep (з обробкою розширених регулярних виразів),
– fgrep (що тлумачить символи $*[]^|()\ буквально),
– rgrep (з включеним рекурсивним пошуком).

2.4.3.5. Технологія відтворення відео файлів

Mpv це програвач, який був створений на базі MPlayer та mplayer2. Плеєр містить вільне та відкрите програмне забезпечення яке включає в себе суміш GNU General Public License версії 2 плюс (GPLv2+), з елементами GNU Lesser General Public License версії 2.1 плюс (LGPLv2.1+) та деякі додаткові частини GNU General Public License версії 3 (GPLv3).

Плеєр працює на кількох операційних системах, включаючи Unix-like версії Berkeley Software Distribution (BSD), Linux, та OS X, а також на Windows. Це багатоплатформна програма, яка працює на ARM, PowerPC, x86 / IA-32, x86-64 та MIPS

mpv мав кілька певних змін з того часу, як його відокремили від MPlayer. Усі видимі частини меню, зі всіма його додатковими функціями, було спрощено та вбудовано у інтерфейс mpv для забезпечення основного контролю за допомогою комп'ютерної миші. Це було зроблено аби поліпшити взаємодію для нових користувачів та впровадити точну і зручну систему пошуку.

Відео сайти: через youtube-dl, mpv природно підтримує високу роздільну здатність (HD) вмісту на YouTube та 300 інших вебсторінок. Це надає

можливість mpv замінювати вбудовані у сайти відео програвачі, які були створені на основі Adobe Flash або HTML5.

Висока якість вихідного відео: mpv включає користувацько-налаштовним відео виводу, який був створений на базі OpenGL. Драйвер підтримує понад 100-ню можливостей для керування якістю відтворення, включаючи використання покращених upscale фільтрів, керування кольорами та користувацько-налаштовними піксель шейдерами[9].

Поліпшений клієнт API: окрім режиму медіа програвача, mpv зроблений для використання іншими програмами напряму, через бібліотеку інтерфейсу, яка має назву libmpv. Це потребує розробку усього mpv коду багатонитково безпечним. Наприклад, програма, що використовує libmpv як Plex[10]. Цей режим керування програвачем, разом з JSON IPC механізмом, заміщує «slave mode» MPlayera.

Підсистема кодування: mpv включає в себе новий режим стиснення даних, який можна використовувати для зберігання відтворюваних на даний момент файлів в різні формати. Це дозволяє mpv працювати як транскодер, підтримуючи багато різних відео форматів. Ця функція послуговує як пряма заміна для додатку MEncoder, що містить у собі MPlayer, який радше був окремою програмою, ніж такою що була вбудована у програвач.

2.5. Опис структури програми та алгоритмів її функціонування

2.5.1. Алгоритм роботи програми для завантаження та перегляду відео

Для завантажування програми використовують більш оптимізовані та стиснуті файли, які містять інформацію, пов’язану з іншими файлами та папками, що підлягають розповсюдженню. Коли користувач завантажує файл фільму, цей "звужений-файл" містить необхідну інформацію, яка дозволить отримати цей фільм та завантажити чи переглянути його.

Розмір файлу не перевищує кількох кілобайт даних. Однак ці крихітні файли можуть розпочати завантаження ще більших файлів, і це робиться через ту чи іншу систему.

Коли починається завантаження файлу можна побачити файл, який не знайомий з цими іншими фрагментами даних. Це значення, які використовуються для перевірки файлу. Коли користувач використовує файл, це майже означає, що він передає файл через мережу P2P або BitTorrent. Під найпростішим поясненням мережа P2P означає мережу, яка створюється, коли два або більше комп'ютерів підключені для спільного використання ресурсів, не проходячи через окремий серверний комп'ютер. Простіше це як спеціальна мережа, але набагато складніша.

Потім файли передаються крихітними бітами, але вони повністю завантажуються у вашу систему після проходження процесу перевірки. Кінцевим результатом є файли, присутні на вашому сховищі, і ось як працює типовий завантажувач.

Ці файли повинні бути доступними повністю від однієї особи або по шматочках, розділених між собою іншими людьми. Щоб сприяти цьому, клієнти працюють на персональній комп'ютерній системі користувача.

По суті, завантажувачі - це міст, який допомагає отримати файли з інших джерел та завантажити їх на свій мобільний пристрій або комп’ютер.

Під час цього процесу користувачу не доведеться якось взаємодіяти з файлами, оскільки сам потік має всю необхідну інформацію, про шматочки файлів "для завантаження", які присутні в різних комп'ютерних системах.

Сам алгоритм завантаження виглядає наступним чином:

1. Користувач відкриває на сторінці посилання, щоб завантажити файл на свій комп’ютер.

2. Програмне забезпечення на вашому комп'ютері (клієнт) повідомляє сервер(центральний комп’ютер, що містить файл, який потрібно завантажити) для передачі копії файлу на ваш комп’ютер.

3. Передача здійснюється за протоколом (набором правил), таким як FTP (Протокол передачі файлів) або HTTP (Протокол передачі гіпертексту).

На швидкість передачі впливає ряд змінних, включаючи тип протоколу, обсяг трафіку на сервері та кількість інших комп'ютерів, які завантажують файл. Якщо файл великий і популярний, вимоги до сервера великі, і завантаження буде повільним.

Завантаження фрагментів файлу одночасно допомагає вирішити поширену проблему з іншими одноранговими методами завантаження: піри завантажують набагато повільніше, ніж вивантажують. Завантажуючи кілька штук одночасно, загальна швидкість значно покращується. Чим більше комп'ютерів бере участь у поширенні, тим швидше відбувається передача файлів, оскільки джерел кожного фрагмента файлу більше. З цієї причини існують користувацькі протоколи, які особливо корисні для великих, популярних файлів.

Протокол - це сценарій, який дозволяє швидко завантажувати великі файли, використовуючи мінімальну пропускну здатність Інтернету. Він максимізує швидкість передачі, збираючи фрагменти потрібного файлу та завантажуючи ці фрагменти одночасно від людей, які їх уже мають. Дуже часто, такі протоколи прописані як взаємодія не тільки з комп’ютерами мережі інших користувачів, а й передбачають наявність іншого «Серверу», який в даному випадку називається трекером. Трекер допомагає клієнтському програмному забезпеченню обмінювати потрібні файли з іншими комп'ютерами в мережі. Тому, комп’ютер отримує кілька фрагментів файлу одночасно.

2.5.2. Алгоритм реалізації програмного завдання Cool Video Downloader

Для успішної реалізації поставлених завдань умовно було складено алгоритм ходу виконання програми (рис. 2.3.).

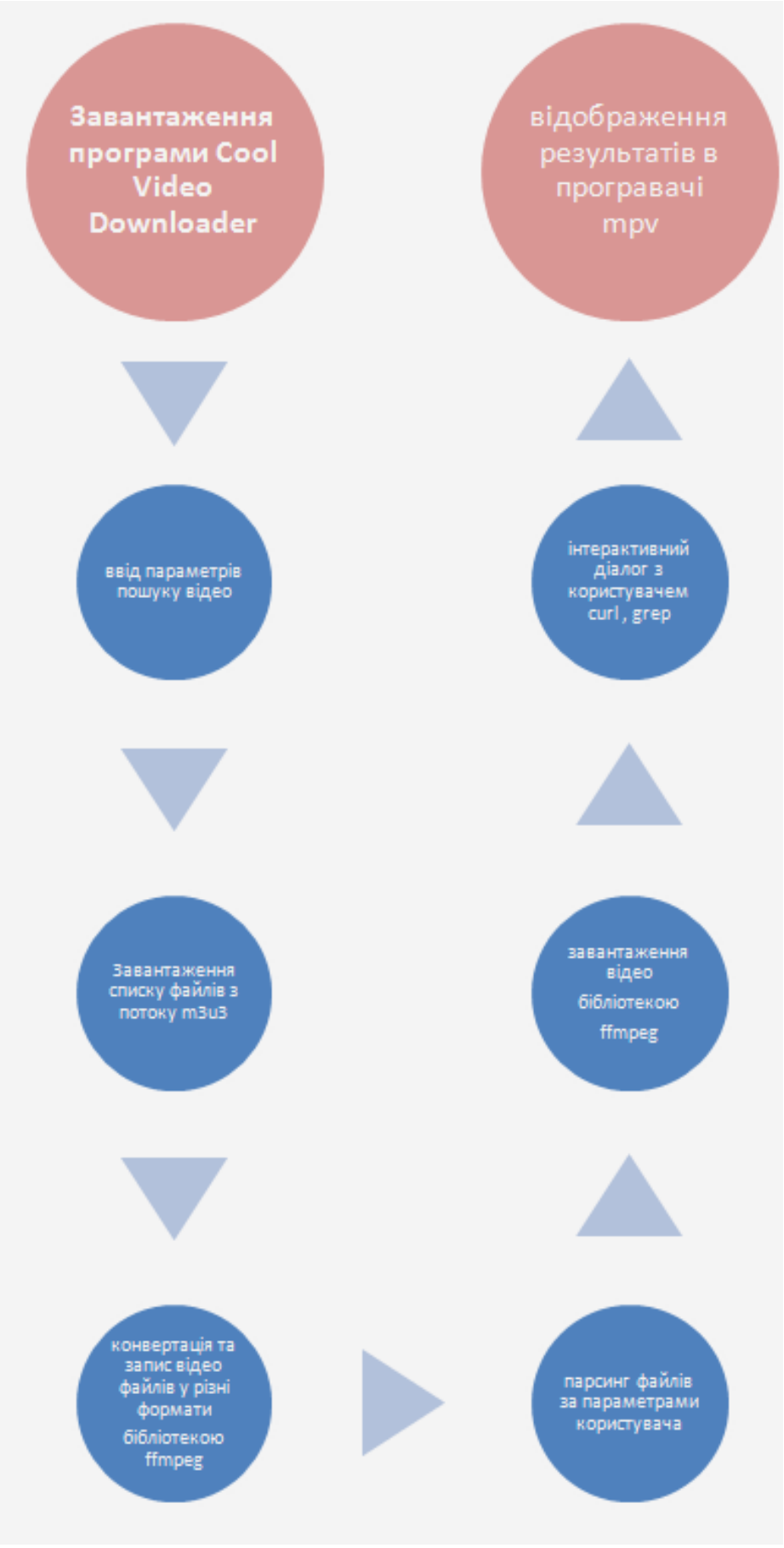


Рис. 2.3. Алгоритм ходу виконання програми

Для завантаження відео з різних ресурсів та його перегляду онлайн без використання браузеру програма виконує наступні етапи:

1. Завантаження додатку Cool Video Downloader.

2. Ввід відповідного посилання на відео файл в текстовий інтерфейс додатку.

3. Програма завантажує m3u8 потік. Цей потік це своєрідний плейлист для програвача. Він в текстовому форматі утримує в собі список мультимедійних файлів для відтворення з точним вказанням шляхів до їхнього розташування на диску чи сервері. Також, там можуть розміщуватись коментарі або інша додаткова інформація. Дані файли застосовуються в багатьох медіаплеєрах.

4. Для успішного завантаження відео, користуючись даною бібліотекою програма проводить ряд конвертацій. Для цього в програмі Cool Video Downloader використовується бібліотека ffmpeg. Вона забезпечує комфортну конвертацію та запис аудіо чи відео файлів у різні формати та є зручною у використанні безкоштовною програмною бібліотекою.

5. Після отримання m3u8 потоку, багатозадачний скрипт виконує процедуру парсингу назв, розширень, номерів серій контенту, який запросив користувач та номери сезону. Ця процедура буде відбуватися напряму від запитів користувача, тому залежить напряму від відповідей сервера. Взагалі, процес парсингу можливий багатьма способами, але так як основне завдання програми все ж у взаємодії з відео контентом, для процедури обробки та систематизування інформації з сайтів було обрано звичайні алгоритми та технології.

6. Так як програма Cool Video Downloader орієнтована на зручність та комфорт у використанні саме для користувача, після запуску скрипта у нього одразу буде запитуватись деяка інформація, яку потрібно ввести до текстового інтерфейсу користувача. Програма очікуватиме будь-який m3u8 потік. Він буде закачаний в звичайний формат відео. У випадку, коли в посиланні буде знайдено щось, що може вплинути на назву файлу, програма автоматично

знайде цей виняток та відповідно змінить назву вихідного файлу. Після цього, відео завантажиться абсолютно без втрат. Для реалізації такого методу було використано бібліотеку ffmpeg.

Ще одним варіантом взаємодії користувача та програми буде те, що в програму можна ввести спеціальне персональне ID потрібного аніме, яке розміщене на сайті anilibria. Знайти його можна на сайті dark-libria.it. З отриманням даного ID Cool Video Downloader буде відтворювати користувацький запит на даний сайт.

Крім того, передбачено також те, що програма може отримувати конкретно посилання на аніме в anilibria. Після цього знову ж таки, Cool Video Downloader буде відтворювати користувацький запит на даний сайт.

7. Після відправки запиту на сервер, програма обробить результат відповіді. Після проведеного парсингу, буде сгенеровано діалог з користувачем, який буде мати алгоритм подальшої взаємодії з програмою. На цьому етапі використовувались бібліотеки такі як: curl, у випадку операційної системи Windows – attrib, ffprobe, grep.

Сам діалог з користувачем відкриває можливість розгалуження подальших дій. Тому, можна завантажити окрему серію сезону медійного контенту, або відкрити відео в mpv. Для реалізації цього використані бібліотеки mpv, ffmpeg, date. Для відкриття відео в mpv текстовий інтерфейс користувача буде очікувати спеціальної команди від користувача. Після введення ключової фрази «mpv» відео відкриється в потрібному плеєрі. Для підтримки по mpv, можна ввести ключову фразу «man mpv». Також, передбачений та реалізований вибір якості відео. Тому, перед командою mpv необхідно ввести цифру. Приклад робочої команди від користувача: «2mpv». Для завантаження абсолютно всього доступного сезону мультимедійного контенту аніме передбачена спеціальна команда – 0. Після неї, розпочнеться завантаження всього сезону аніме.

2.6. Обґрунтування та організація вхідних та вихідних даних програми

У якості вхідних даних виступає введена у поле пошуку назва, розширення, номери серій контенту сезонів відео для завантаження його в додаток.

Вихідними даними є отримана інформація та відображення обраного відео файлу у програвачі.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для розробки даного програмного продукту використовувався ПК з наступними характеристиками:

– CPU 1.6 Hz;
– VGA 512 Mb;
– RAM 1 Gb;
– монітор з роздільною здатністю 1680/1050;
– клавіатура;
– миша;
– доступ в мережу Інтернет.

2.7.2. Використані програмні засоби

Для розробки програми Cool Video Downloader була застосована мова програмування Bash.

2.7.3. Виклик та завантаження програми

Bash - це оболонка і командна мова Unix, який може запускати файли скрипти .sh.

Не потрібно встановлювати Ubuntu або будь-які інші дистрибутиви Linux, якщо скрипти не потребують підтримки реального ядра Linux.

Для першого запуску програми в Bash можна використати WSL:

1. Встановіть WSL або Windows Subsystem для Linux

2. Перейдіть в Установки> Оновлення та безпека> Для розробників.

3. Перевірте перемикач режиму розробника.

4. Пошукайте «Функції Windows», виберіть «Включити або відключити функції Windows».

5. Прокрутіть, щоб знайти WSL, встановіть прапорець і встановіть його.

6. Після цього необхідно перезавантажити комп'ютер, щоб завершити установку запитаних змін. Натисніть Перезавантажити зараз.

BASH відтепер буде доступний в командному рядку і PowerShell.

Для виконання файлу скрипта оболонки:

1. Відкрийте командний рядок і перейдіть в папку CVD, де доступний файл скрипта.

2. Введіть Bash script-CVD.sh і натисніть клавішу Enter.

3. Система виконає скрипт, і почнеться робота програми.

2.7.4. Опис інтерфейсу користувача

Для повного тестування розробленої програми, потрібно проаналізувати та перевірити всі заявлені функції. Після реалізації скрипта, можна запустити його та спробувати переглянути чи завантажити потрібне аніме.

Для того, аби завантажити потрібні нам серії аніме, знаходимо відповідне посилання на нього в мережі інтернет. Потім, вводимо це посилання в

текстовий інтерфейс програми, аби вона продовжила обробку даного запита користувача.

Якщо все пройде успішно, скрипт відобразить інформацію про статус аніме та продовжить діалог з користувачем (рис. 2.4). Програма запропонує обрати якість відео та почне завантажувати відео файл.

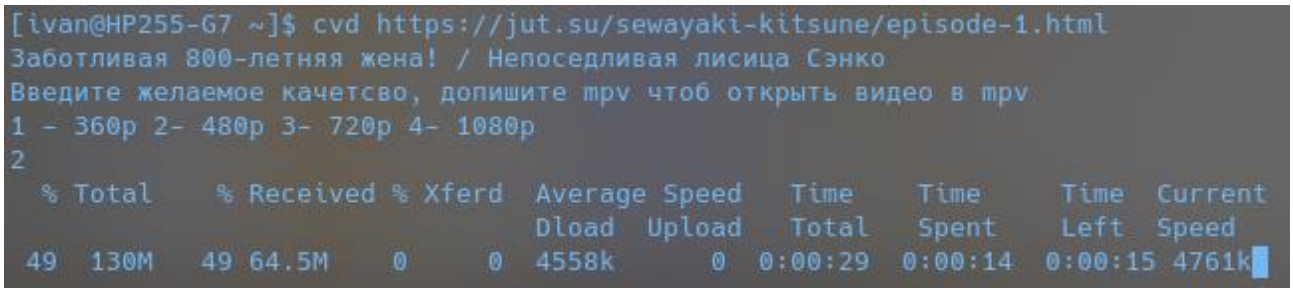


Рис. 2.4. Зображення процесу завантаження за заданим посиланням з платформи jut.su

Після перевірки цілісності та відповідності посилання, програма обробить отриману відповідь та виведе подальші кроки в діалозі з користувачем, де наступний зможе вибрати кількість серій та якість відео. В результаті виборів почнеться завантаження серій (рис. 2.5).

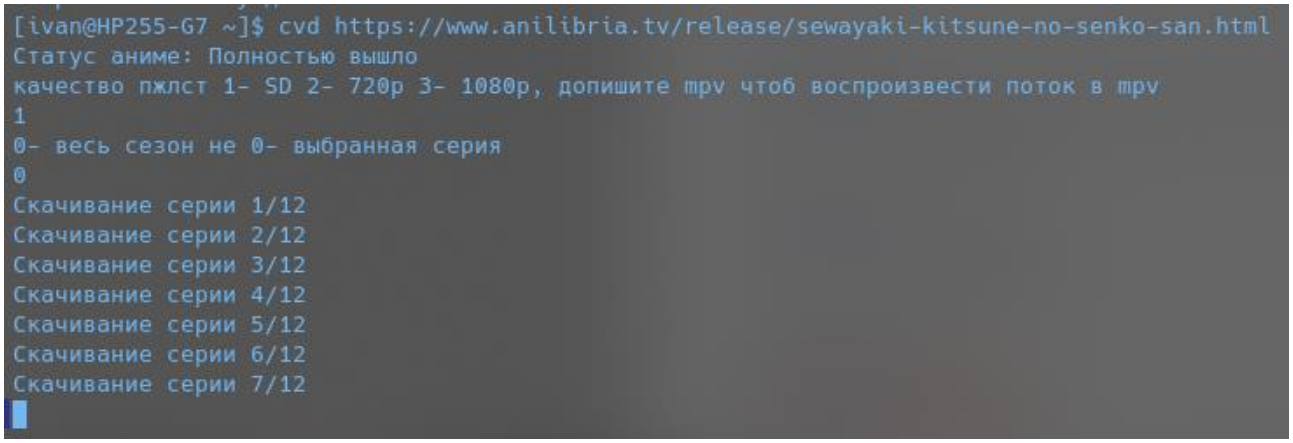


Рис. 2.5. Процес скачування медіа файлів за обраними параметрами

Також, програмою передбачений онлайн перегляд контенту аніме. Для цього, потрібно ввести посилання та відповідно до команд, почнеться

відтворення у вибраному плеєрі та вибраній якості. На рис. 2.6 зображений приклад такого введення та результати діалогу з користувачем.

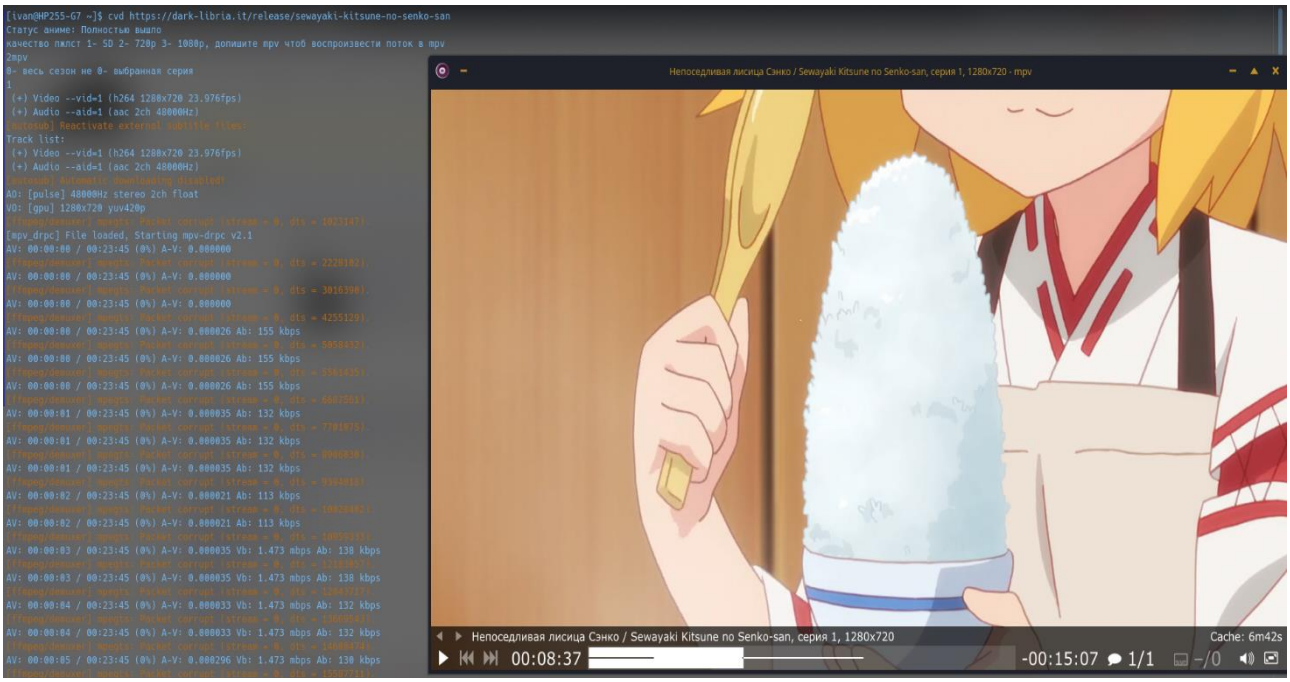


Рис. 2.6. Онлайн транслювання відео за допомогою розробленої програми Cool Video Downloader

Також на рис. 2.6 можна чітко простежити пакетування та всі необхідні часові інтервали. По деяким причинам, пакети з anilibria завжди помічаються як пошкоджені. Однак завантаження та відтворення проходить без збоїв, а самі пакети цілком цілі та відтворюються відповідно поставленій меті.

Програма працює без збоїв та незапланованих завершень. Завдяки підключенню стандартного плеєра, онлайн відтворення можна зупинити, прогорнути або ж почати спочатку, що тільки підсилює комфорт користувача. Також, для зручності, текстовий інтерфейс сповіщає про абсолютно усі виконувані процеси програмою. Це дуже важливо для правильного розуміння та успішної експлуатації програми користувачем. Звичайно, в майбутньому для більшого комфорту можна буде розробити та реалізувати власний інтерфейс програми, який буде ще більш чітко відображати всі дії скрипта та його взаємодію з користувачем (рис. 2.7).

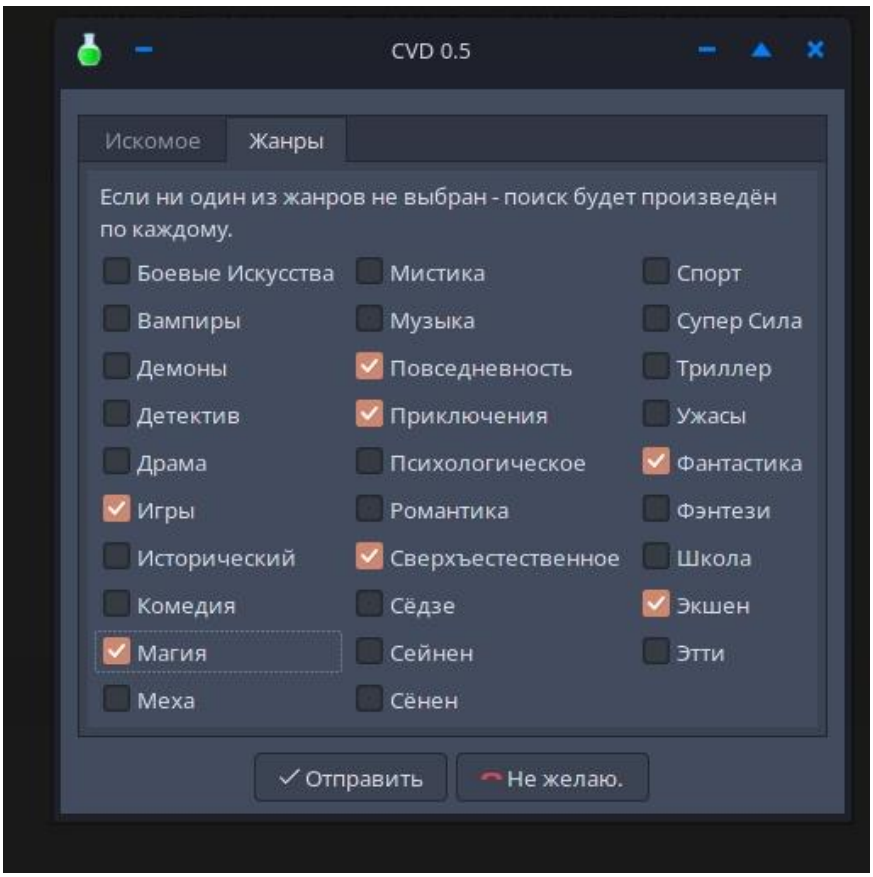


Рис. 2.7. Приклад взаємодії програми з користувачем.

Після завершення повного тестування програми критичних багів виявлено не було, а тому продукт є повністю функціональним та багатозадачним, так як за допомогою потоку, вона підтримує скачування з торрент трекерів, YouTube, а також передбачає підставку озвучення відео контенту з одного ресурсу на інший з розрахунком затримки.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів - 1500;
- коефіцієнт складності програми - 2;
- коефіцієнт корекції програми в ході її розробки - 0,08;
- годинна заробітна плата програміста, грн./год - 100.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де to – витрати праці на підготовку й опис поставленої задачі (50),

ти – витрати праці на дослідження алгоритму рішення задачі,

та – витрати праці на розробку блок-схеми алгоритму,

тп – витрати праці на програмування по готовій блок-схемі,

тотл – витрати праці на налагодження програми на ЕОМ,

тд – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \quad (3.2)$$

де q - передбачуване число операторів,

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

$$Q = 1500 \cdot 2 \cdot (1 + 0{,}08) = 3240 \text{ людино-годин.} \quad (3.3)$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K}, \text{ людино-годин,} \quad (3.4)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; B=1.2 … 1.5,

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{3240 \cdot 1{,}3}{80 \cdot 1{,}2} = 44, \text{ людино-годин.} \quad (3.5)$$

Витрати на складання програми по готовій блок-схемі:

$$t_a = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \quad (3.6)$$

$$t_a = \frac{3240}{22 \cdot 1{,}2} = 123 \text{ людино-годин.} \quad (3.7)$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.8}$$

$$t_n = \frac{3240}{23 \cdot 1{,}2} = 117 \text{ людино-годин.} \tag{3.9}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.10}$$

$$t_{отл} = \frac{3240}{5 \cdot 1{,}2} = 540 \text{ людино-годин.} \tag{3.11}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.12}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.13}$$

$$t_{др} = \frac{3240}{18 \cdot 1{,}2} = 150 \text{ людино-годин,} \tag{3.14}$$

де tдо - трудомісткість редагування, печатки й оформлення документації.

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \tag{3.15}$$

$$t_{до} = 150 + 73 = 223 \text{ людино-годин.} \tag{3.16}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 44 + 123 + 117 + 540 + 223 = 1097 \text{ людино-годин.} \tag{3.17}$$

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.18)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.19)$$

де t - загальна трудомісткість, людино-годин,

Спр - середня годинна заробітна плата програміста, грн/година.

$$Ç_{ÇÏ} = 1097 \cdot 100 = 109700 \text{ ãðí.} \quad (3.20)$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times C_{М}, \text{ грн,} \quad (3.21)$$

де tотл - трудомісткість налагодження програми на ЕОМ, год.,

Смч - вартість машино-години ЕОМ, грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 540 \times 5 = 2700 \text{ грн.}, \quad (3.22)$$

$$Êïï = 2700 + 109700 = 111400 \text{ ãðí.} \quad (3.23)$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.,} \tag{3.24}$$

де Bk - число виконавців,

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{1097}{1 \cdot 176} = 6.2 \text{ міс.} \tag{3.25}$$

Визначено трудомісткість розробленої інформаційної системи (1097 люд-год), проведений підрахунок вартості роботи по створенню програми (111400 грн/) та розраховано час на його створення (6,2 міс).

ВИСНОВКИ

В даній кваліфікаційній роботі було виконано та реалізовано повнофункціональну програму, що містить скрипт для закачування та онлайн перегляду відео файлів.

Робота була реалізована за допомогою технології Bash.

Під час виконання даного завдання було виконано загальну характеристику, опис особливостей програмування та технологічного забезпечення даної проограми.

Згідно завдань було здійснено:

1. Детальне ознайомлення з принципами та особливостями роботи певних технологій та компонентів, що використовувались у процесі розробки програми.

2. Було створено та детально описано процес написання програмного продукту. Також, вказано особливості створення програми Cool Video Downloader та правильне її завантаження.

3. Протестовані результати роботи скрипта та опис діалогу з користувачем.

Програма працює без збоїв та незапланованих завершень. Завдяки підключенню стандартного плеєра, онлайн відтворення можна зупинити, прогорнути або ж почати спочатку, що тільки підсилює комфорт користувача. Також, для зручності, текстовий інтерфейс сповіщає про абсолютно всі виконувані процеси програмою. Це дуже важливо для правильного розуміння та успішної експлуатації програми користувачем.

В майбутньому для більшого комфорту можна буде розробити та реалізувати власний інтерфейс програми, який буде ще більш чітко відображати всі дії скрипта та його взаємодію з користувачем.

Готовий продукт є повністю робочим та може використовуватись для зручної взаємодії користувача з сервісами надання доступу до відео, зокрема аніме, без відкриття браузера. Даний скрипт вирішить проблему користувачів,

які мають регулярні збої в мережі Інтернет та відкриє доступ до зручного та масового завантаження потрібного відео контенту.

В економічному розділі визначено трудомісткість розробленої інформаційної системи (1097 люд-год), проведений підрахунок вартості роботи по створенню програми (111400 грн/) та розраховано час на його створення (6,2 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Блек Ю. Сети ЭВМ: Протоколы, стандарты, интерфейсы: Пер. с англ. – М.: Мир, 1987. – 356 с.

2. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).

3. Бусигін Б.С., Коротенко Г.М., Коротенко Л.М. Прикладна інформатика. Підручник для студентів комп'ютерних спеціальностей. – Дніпропетровськ: Видавництво НГУ, 2004. – 559 с. URL: http://www.programmer.dp.ua/book-ua-k01.php. дата звернення: 22.02.2021.

4. Бусыгин Б.С., Дивизинюк М.М., Коротенко Г.М., Коротенко Л.М. Введение в современную информатику. Учебник. – Севастополь: Издательство СНУЯЭиП, 2005. – 644 с. / URL: http://www.programmer.dp.ua/book-ru-k02.php. дата звернення: 15.01.2021.

5. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

6. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.

7. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.

8. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 121 «Програмна інженерія» галузі знань 12

Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2019. – 65 с.

9. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

10. Основы BASH. Часть 1 URL: https://habr.com/ru/post/47163/.

11. Программы для скачивания видео с интернета URL: https://soft.mydiv.net/win/collections/show-Programmy-dlya-skachivaniya-video-s-interneta.html.

12. Статистика YouTube 2019. Инфографика URL: https://exlibris.ru/news/statistika-youtube-2019-infografika/ дата звернення: 22.02.2021

13. Bash – командная оболочка Linux [URL: https://younglinux.info/bash/terminal.php. дата звернення: 22.02.2021

14. Bash-скрипты: Руководство по Функциям Bash с Примерами URL:https://www.hostinger.com.ua/rukovodstva/bash-skripty-rukovodstvo-po-funkcijam-bash-s-primerami/. дата звернення: 22.02.2021

15. Introduction to the Bash Command Line URL: https://programminghistorian.org/en/lessons/intro-to-bash. дата звернення: 22.02.2021

16. Lottor M. K. Request for comments: 913. Simple File Transfer Protocol. – MIT: Network Working Group, 1984.

17. Postel J., Reynolds J. Request for comments: 959. File Transfer Protocol. – ISI: Network Working Group, 1985.

18. Postel J., Reynolds J. Request for comments: 854. Telnet Protocol Specification. – ISI: Network Working Group, 1983.

19. What is SDLC? Phases of Software Development, Models, & Best Practices URL: https://phoenixnap.com/blog/software-development-life-cycle. дата звернення: 22.02.2021

20. What are the Software Development Life Cycle (SDLC) phases? URL: https://www.linkedin.com/pulse/what-software-development-life-cycle-sdlc-phases-private-limited/. дата звернення: 22.02.2021

ДОДАТОК А

КОД ПРОГРАМИ

```
#!/bin/bash
#cvd alpha 0.5 by "Здесь ты ищешь тот шар золотой..."

start=$(date +%s) # требуеться для посчёта времени затраченного на выполнение
yad=1 # GUI

if [ $yad = 1 ]; then
curl https://dark-libria.it/ -o .cvd.dark.tmp 2> .cvd.curl.tmp
gcut=$(echo $(cat .cvd.dark.tmp) | sed 's/.*genre//g')
genres=$(echo $gcut | sed 's/<\/select>.*//g' | sed s/'<\/option> <option>'/':chk" --field=$"'/g | sed
s'/<\/option>'/':chk"'/g | sed s/'" size="6" data-live-search="true"> <option>'/'--field=$"'/g);while [[
"$genres" = *DOCTYPE* ]]; do genres=$(echo $(curl https://dark-libria.it/ -o .cvd.dark.tmp 2>
.cvd.curl.tmp;cat .cvd.dark.tmp;gcut=$(echo $(cat .cvd.dark.tmp) | sed 's/.*genre//g')) | sed
's/<\/select>.*//g' | sed s/'<\/option> <option>'/':chk" --field=$"'/g | sed s'/<\/option>'/':chk"'/g | sed
s/'" size="6" data-live-search="true"> <option>'/'--field=$"'/g);
done
echo yad --plug=$start --tabnum=2 --form --columns=3 $genres --text=\"Если ни один из жанров
не выбран - поиск будет произведён по каждому\". \> .cvd.tf.tmp > yad-alert;bash yad-alert &

fi
argsV='-y -c copy -bsf:a aac_adtstoasc' # аргументы для скачивания видео (только ffmpeg)

argsA='-y -bsf:a aac_adtstoasc' # аргументы для скачивания аудио, впервые появились в альфа
pre-0.5, пока им применение не найденно
link=$@ #

cont=mkv # контейнер для скачанных файлов
downtime=10 # задежка между попытками скачать новую серию
debug=0 # debug level
gospod='Чего желаете, господин?' # входное сообщение
cvdw=203 # Ширина маленьких окон
```

```
progname='CVD 0.5' # кодовое название
#mpvargs='--hwdec=auto'
if [ -z "$2" ]; then name=video # если скачиваться видео со страшного ресурса, то его
                                else name=$2 #    имя будет вторым аргументом, если не указан -
video
fi
if [ -z "$link" ] ; then
                                        until [ "$err" = 0 ]; do


        #
                                        if [ $yad = 1 ]; then yad --entry --text="$gospod" --
width=$cvdw  --plug=$start --tabnum=1 > .cvd.link.tmp &
                                        yad --notebook --key=$start --tab=$"Искомое" --
tab=$"Жанры" --title="$progname" --buttons-layout=center --button='Отправить!answer:0' --
button='Не желаю.!call-stop:1' --text-align=center --center
                                        tf=$(cat .cvd.tf.tmp)
                                        link=$(cat .cvd.link.tmp)
                                        all=$(echo "$tf" | grep -o "|" | wc -l)
tfr=$(echo $gcut | sed 's/<\/select>.*//g' | sed s/'<\/option> <option>'/\|/g | sed s'/<\/option>'/'|'/g | sed
s/'" size="6" data-live-search="true"> <option>'/"/g)
                                        if [ -z $tf ]; then :;else
until [ "$temp" = "$all" ]
do
let 'temp = temp + 1'
if [ $(echo $tf | cut -d\| -f$temp) = TRUE ];
 then extra=$extra\&genres\=$(echo $tfr | cut -d\| -f$temp);fi
done
fi

                #приветствие.
                                        else
                                        echo $gospod #
        консольное    #приветствие.
```

```
                                        read link # считыватель введенных аргуме.. A
REDACTED его,                                   fi
                                    link=$link | sed 's/^[ \t]*//;s/[ \t]*$//' | cut -d? -f1 #
                        if [ -z "$link" ];
then
                                    err=1;
gospod=умный? введи чёт.' #
                                    if [ $yad = 1 ];then exit;fi
                                    else err=0
                                    fi
                                                        done
fi
if [ "$link" = q ] && [[ $@ != q ]] ;
 then
echo Выход...;
exit # выход при вводе q
   elif [[ "$link" = http*://*.libria.fun/videos/ts/* ]] ;
then                                                #доREDACTEDа
          num=$(( ${#link} - 20 ))
                  #умный считатор
    num=$(echo $link | cut -c$num-$(let 'num = num + 6'; echo $num))    #номера
          if [[ "$num" = */* ]] ;
 then
num=$(echo $num | cut -c4-7)-HD              #для
    elif [[ "$num" = *-sd ]] ;
then
num=$(echo $num | cut -c1-4)-360p #анилибры
    elif [[ "$num" = *-hd ]] ;
then num=$(echo $num | cut -c1-4)-1080p   #
          elif [ "$err" != 0 ]; then
          echo Неизвестная ошибка связанная с получением качества аниме!                    #
          echo Прошу, напишите мне,
   fi
          echo Скачивание началось! ожидайте..
          trap "int=1;
```

```
echo Прерванно." INT
        ffmpeg -i $link $argsV $num.$cont #2> .cvd.tmp       устарело
        if [ "$int" = 1 ];
then
 rm -rf $num.$cont;exit                    #в
        else
 exit;fi # скачивалка, аргументы идут в .cvd.tmp  0.5
        elif [[ "$link" = *.m3u8 ]] ;
 then # хандлер m3u8 потоков
         ffmpeg -i $link $argsV $name.$cont # скачивалка
        elif [[ $link = *dark-libria.it/release/* ]];
then #ОБНАРУЖЫЖЕВАТИТИВАТЕЛЬ анилибрЫ
        findby=code;
err=0;
listing=anilibria
        link=$(echo $link | cut -d: -f2)
        if [[ $link = //* ]];
then link=$(echo $link | cut -c26-)
        else
 link=$(echo $link | cut -c24-)
        fi
        elif [[ $link = *anilibria.tv/release/* ]];
then #тоже ОБНАРУЖЫЖЕВАТИТИВАТЕЛЬ анилибрЫ
        findby=code;
err=0;
listing=anilibria
        link=$(echo $link | cut -d: -f2)
        if [[ $link = //* ]];
 then link=$(echo $link | cut -c28- | cut -d. -f1)
        else
link=$(echo $link | cut -c24-)
        fi
         elif (echo "$link" | grep -E -q "^?[0-9]+$") && [ ${#link} = 3 ] || [ ${#link} = 4 ] && (echo
"$link" | grep -E -q "^?[0-9]+$");
```

```
          then
        # введён айдишник или нет
         if (( $debug > 2 ));
then echo reached target 2;fi
          findby=id;
err=0;listing=anilibria # прадед, сыщик анилибры.
          if (( debug > 1 ));
then echo $link has been used.;fi
          fi
        if [ -z $listing ];
then
        codeplus=$(echo $link | sed s/' '/+/g | sed s/-/+/g)                    temp=$(curl
https://dark-libria.it/search?find=$codeplus$extra 2> .cvd.curl.tmp | grep -i '<a href="/release/' | cut
-d"" -f2 | cut -d\/ -f3)
        if [ -z "$temp" ];
 then :
        else link=$temp
        listing=anilibria;
findby=code;
err=0
        fi;fi
                jutlink=$(curl -Lf https://jut.su/search -F ystext=$link -F makeme=yes 2>
.cvd.curl.tmp | sed s/'<a itemprop="item" href="https:\/\/jut.su\/anime\/">'/"/g | grep -i '<a
itemprop="item" href="' | cut -d"" -f4)
                if [ $jutlink = 'https://jut.su/search/' ];
then jutlink='';
fi #рандомная
                #echo $jutlink
        if [[ $link = *jut.su/* ]];
then listing=jutsu;fi
        if [ "$listing" = anilibria ];
 then
          if [ "$err" = 0 ];
then
          until [ -f .cvd.tmp ]
```

```
            do
                touch .cvd.tmp
                if [[ ~ = /c/Users/* ]];
then attrib +S +H .cvd.tmp > /dev/null
                fi
                if [ -f .cvd.tmp ] ;
then :
                else
                    if [ -f ~/Загрузки ] ;
                        then
                        echo Переход в загрузки...
                        cd ~/Загрузки
                        elif [ -f /sdcard/Downloads ];
                        then
                        echo Переход в загрузки...
                        cd /sdcard/Downloads
                        else
                        echo Changing dir to downloads...
                        mkdir -p ~/Downloads
                        cd ~/Downloads
                    fi
                  fi
                done
              id=$link
        if [ $yad = 0 ];then printf "Отправка запроса.."
        #else echo "# Отправка запроса"
        fi
        resp=$(curl -Lf 'https://www.anilibria.tv/public/api/index.php' -F query=release -F
$findby=$link 2> .cvd.tmp)
        fi
        if [[ $resp = *status*false*data*null*error*code*404*message*Release*by*$findby=* ]] ;
then
         rm -rf .cvd.tmp
    printf "\rОшибка 404! Неверный $findby! Аргументы сброшены! Перезапуск скрипта..
"
```

```
        $0
        exit
        elif [[ $(cat .cvd.tmp) = *curl*35*443* ]];
        then            #Quichi, спасибо за тестирование
        echo 'Ошибка curl (35), вероятнее всего в вашей стране (Чаще всего Росии)
заблокиорван домен www.anilibria.tv (104.27.195.89)'
        echo попробуйте ещё раз с VPN настроеным не на Россию.
        exit
        elif [[ $(cat .cvd.tmp ) = *curl*6*resolve ]];
 then
        echo 'Ошибка curl (6), проверьте доступ в интернет.'
        exit
        fi
        echo $resp | jq 'if .status then .data else error(.error) end' > .cvd.tmp
        nums=$(grep series .cvd.tmp);nums=$(echo $nums | cut -f1 -d,)
        first=$(echo $nums | cut -f1 -d-)
        allnum=$(grep type .cvd.tmp | cut -f2 -d'(' | cut -f1 -d' ')
                        if [[ $allnum = *эп.* ]];
then
                                                allnum=$(echo $allnum | cut -f1 -
d.);temp=${#allnum};let 'temp = temp -2';allnum=$(echo $allnum | cut -c1-$temp) #               fi
        #fhd=$(grep 'hd/playlist.m3u8' .cvd.tmp)
        lfhd=$(grep -i fullhd .cvd.tmp | tac | cut -d'
' -f1 | cut -c56-62 | sed s/-/''/g | sed s/h/''/g | sed s/d/''/g)
                                                                        if [ ${#lfhd} = 0 ];
then lfhd=0
                                                                                fi
        ll=$(grep playlist.m3u8 .cvd.tmp);
ll=$(echo $ll | cut -f2 -dp | cut -f1 -dv | cut -c5-${#ll} 2>> .cvd.tmp)
        if [[ $(cat .cvd.tmp) = *cut*--help* ]];
        then
        #if [ -есть) .cvd.tmp ];
 then echo 'отвевет не получен! ошибка #1'
        #else
        echo "Неизвестная ошибка связанная с получением ответа! #2"
```

```
        #fi
        echo Прошу, напишите мне, ХАЛАДОС#8335
        $0;
exit
        fi
        if [ ${#first} = 12 ] ;
then first=$(echo $first | cut -c 12)
         else first=$(echo $first | cut -c 12-${#first} | cut -f1 -d"")
        fi                                      # а эта REDACTEDня спасёт, если серий одна
        if [[ "$nums" != *-* ]] ;
then
        last=$first
   else
        last=$(echo $nums | cut -d- -f2)
        temp=${#last};let 'temp = temp - 1'
        last=$(echo $last | cut -c 1-$temp)
        num=$(echo 000$first | cut -c${#first}-7)
        fi
                        if [[ $(grep status\" .cvd.tmp) = *авершен* ]]; then allnum=$last;
status=ready;fi
        if [[ "$allnum" == [\>]* ]];
 then #5614
                typehaserr=">"
                allnum=???
        fi

                                        #king039s-raid-ishi-wo-tsugumono-tachi
        names=$(echo $(echo $resp | cut -d[ -f2 | cut -d] -f1) | sed s/"","'/' \/ '/g | cut -c 2- | rev | cut -c
2- | rev | sed 's/^[ \t]*//;s/[ \t]*$//' | sed s/'&#'/"/g)
        if [ $findby = code ]; then findby=id;
id=$(echo $resp | cut -d, -f2 | cut -c14-);fi

        shitnum=$(cat .cvd.tmp | grep -i series | cut -d'
' -f2 | rev | cut -c3- | cut -d"" -f1 | rev)
        if [ "$shitnum" != $first-$last ] && [ $first != $last ];
```

```
 then
        wtf=1
        if ((debug > 0));
 then echo shit heppens.;fi
        let 'shitometr = first - 1'
        rm -rf .cvd.non.tmp
        until [ "$done" = yes ]; do
        #echo $shitometr
        numshit=000$shitometr;numshit=$(echo $numshit | cut -c${#shitometr}-7)
        #grep $num-sd/playlist.m3u8 .cvd.tmp
        if [ -z "$(grep $numshit-sd/playlist.m3u8 .cvd.tmp)" ];
then
echo .$shitometr. >> .cvd.non.tmp
        fi
        let ' shitometr = shitometr + 1 '
        #echo $shitometr govno
                if [ "$shitometr" = "$last" ];
                        then
                        done=yes
                fi
        done
        fi
        #description=$(grep description .cvd.tmp | cut -d"" -f4)
        description=$(grep description .cvd.tmp | cut -c19- | rev | cut -c3- | rev | sed s/'<br>'/'\n'/g |
sed s/'&gt;'/'>'/g | sed s/'<a href=\\"'/''/g | sed s/'<\/a>'/''/g | sed s/'\\" target=\\"_blank\\">'/' '/g)
        if [ $yad = 1 ];
 then
        temp=$(xdpyinfo | grep 'dimensions:' | cut -d' ' -f7);yad --no-buttons --title=Описание --text
"$description" --show-cursor --show-uri --width=$(( $( echo $temp | cut -d'x' -f1) / 3 )) --posx=$((
$( echo $temp | cut -d'x' -f1) / 24 )) --posy=$(( $( echo $temp | cut -d'x' -f2) * 2 / 7 )) --on-top --
skip-taskbar &
        fi
        if [ $yad = 0 ];
then
        printf "\rСтатус аниме: "
```

```
        if [ "$status" = ready ];
then echo Полностью вышло
        else
        echo в работе
        fi
        printf Названи
                if [[ "$names" = */* ]];
then
                echo я: $names
                else
                echo e: $names
                fi
        if (( debug > 1 ));
 then echo $resp;  fi
        else
                if [ "$lfhd" != 0 ];
then h=157;
shit='1080p' # 168
                hdname='720p'
#               sdname='360p'
                else h=132 #144
                hdname='HD'
#               sdname='SD'
                fi
                temp=err
                temp=$(yad --list --title="$progname" --column="Выбор графония" 360p $hdname
$shit --text "выберите графоний" --hide-header --height=$h --text-align=center --center --
width=$cvdw --buttons-layout=center)
                if [ -z $temp ];
then exit;fi
                if yad --question --title="$progname" --text="Открыть поток в mpv?" --
button='Да!answer:0' --button='Нет!download' --text-align=center --center --width=$cvdw --
buttons-layout=center;then mpv=1;fi
                if [ "$temp" = '720p|' ] || [ "$temp" = 'HD|' ] ; then temp=2;fi
#               if [ "$temp" = '360p|' ] || [ "$temp" = 'SD|' ] ; then temp=1;fi
```

```
                if [ "$temp" = '360p|' ]; then temp=1;fi
                if [ "$temp" = '1080p|' ] ; then temp=3;fi
        fi
        elif [ "$listing" = jutsu ];
 then
        if [ $yad = 0 ]; then
        printf "Скачивание страницы.."
        else :
        fi
        #(
        #echo "10"
        #echo "# Скачивание Страницы"
        curl $link -A "" 2> .cvd.curl.tmp | iconv -f cp1251 > .cvd.tmp
        if [ -z "$(grep '<title>404 Not Found</title>' .cvd.tmp)" ];
 then :
        else echo '
Неверная ссылка! 404!';
echo Перезапуск..;$0;
exit;fi
        if [[ "$link" != *episode-*.html ]];
 then
        jutlast=$(cat .cvd.tmp | grep -i '</i>' | grep -i Серия | rev | cut -d' ' -f2 | cut -d'>' -f1 | rev)
        first=$(cat .cvd.tmp | grep -i '</i>' | grep -i Серия | cut -d'>' -f4 | cut -d' ' -f1)
        names=$(grep '<meta property="yandex_recommendations_title" content="' .cvd.tmp | sed
s/'<meta property="yandex_recommendations_title" content="'/"/g | rev | cut -c5- | rev)
        #names='ЮЮЮЮЮЮЮ'
        code=$(grep '<link rel="canonical" href=' .cvd.tmp | cut -d"" -f4 | cut -c16- | cut -d/ -f1)
        codeplus=$(echo $code | sed s/-/+/g)
        temp=$(grep '<meta itemprop="alternateName" content="' .cvd.tmp | sed s/'<meta
itemprop="alternateName" content="'/"/g | rev | cut -c4- | rev)
        w=$(( ${#names} * 11 + 170 ))

        #echo $w
        #echo ${#names}
```

```
	if [ $yad = 1 ]; then num=$(yad --scale --min-value=1 --title="$names" --max-value=$jutlast --text='Выбор серии' --min-value=$first --value=$first --text-align=center --center --width=$w --buttons-layout=center)
	else printf "\rВведите номер серии от $first до $jutlast:"
	read num
	fi
	curl $link/episode-$num.html -A "" 2> .cvd.curl.tmp | iconv -f cp1251 > .cvd.tmp
	fi
	link=$(grep '<link rel="canonical" href=' .cvd.tmp | cut -d"" -f4)
	lfhd=$(cat .cvd.tmp | grep -i .1080 2>> .cvd.tmp| cut -c14- | cut -d"" -f1 | cut -d' ' -f1)
	names=$(grep '<h1 class="header_video allanimevideo the_hildi anime_padding_for_title_post">' .cvd.tmp | sed s/'>'/"/g | sed s/'<i'/"/g | cut -d'<' -f4 | sed s/'\/i'/"/g)
	#echo "# Получение альтернативных названий"
	#echo "70"
	if [ $yad = 0 ]; then printf "\rПолучение альтернативных названий..";fi
	temp=$(curl https://dark-libria.it/search?find=$codeplus 2> .cvd.curl.tmp | grep -i "</div>'><span>" | cut -d'>' -f4 | cut -d'<' -f1 &)
	season=$(grep '<span itemprop="name">' .cvd.tmp | grep -i сезон | cut -d'>' -f2 | cut -d' ' -f1)
	num=$(grep '<link rel="canonical" href=' .cvd.tmp | rev | cut -c10- | cut -d- -f1)
	#echo "100"
	#)  |
	#if [ $yad = 1 ];
then yad --progress --title="Парсинг.." --percentage=0 --auto-close --no-cancel;fi
	if [ -z $lfhd ];
then lfhd=0;fi
	hd=$(cat .cvd.tmp | grep -i .720 2>> .cvd.tmp| cut -c14- | cut -d"" -f1 | cut -d' ' -f1)
	if [ -z "$hd" ];
then hd=0;fi
	notso=$(cat .cvd.tmp | grep -i $num.480 2>> .cvd.tmp| cut -c14- | cut -d"" -f1 | grep -i .480. | cut -d';' -f2 | sed 's/^[ \t]*//;s/[ \t]*$//')
	sd=$(echo $(cat .cvd.tmp | grep -i $num.360 2>> .cvd.tmp| cut -c14- | cut -d"" -f1 | sed s/pe=/"/g | sed s/ass=/"/g))
	if [ -z $num ];
```

```
then num=$(echo $sd | sed s/$season/"/g 2>> .cvd.tmp | sed s/'\/\/'/"/g | sed s/$code/?/g | cut -d? -f2 |
cut -d. -f1);fi
        if [[ $names = *ерия* ]];
then names=$(echo $names | cut -d"$num" -f1);fi
        names=$(echo $names)
        if [ -z "$names" ];
then names=$(grep '<span itemprop="name">' .cvd.tmp | grep -i Смотреть | cut -d'<' -f3 | sed
s/'\/span>'/"/g | cut -d'>' -f2 | sed 's/Смотреть/"/g;s/$num\ серия/"/g;s/$season\ сезон/"/g;s/^[
\t]*//;s/[ \t]*$//;s/[:\t]*$//');fi
        if [ -z "$temp" ];
 then :;else names=$(echo $names / $temp);fi
        printf "\r$names"
        fi
        if [ $yad = 0 ];
then
        echo -e Описание: $description
        fi
        until [[ $err = 'no' ]]
        do
        if [ "$listing" = anilibria ];
then
                if [ "$lfhd" != 0 ];
 then echo качество пжлст 1- 360 2- 720p 3- 1080p, допишите mpv чтоб воспроизвести поток в
mpv
                else
echo качество пжлст 1- SD 2- HD, допишите mpv чтоб овспроизвести поток в mpv
                fi
        elif [ "$listing" = jutsu ];
then
        if [ $yad = 0 ];
then echo "
Введите желаемое качетсво, допишите mpv чтоб открыть видео в mpv"
        printf '1 - 360p 2- 480p'
        if [ "$hd" != 0 ];
then printf ' 3- 720p';fi
```

```
        if [ "$lfhd" != 0 ];
 then echo ' 4- 1080p';
else echo ";fi
        else
        h=134
        if [ "$hd" != 0 ];
then hdkoldun='720p';let 'h = h +24';fi
        if [ "$lfhd" != 0 ];
then shit='1080p';let 'h = h +24';fi
        echo $h
        temp=$(yad --list --title="$progname" --column="Выбор графония" "360p" "480p"
$hdkoldun $shit --text "выберите графоний" --hide-header --height=$h --center --text-align=center
--width=$cvdw --buttons-layout=center)
        if [ -z $temp ];
then exit;fi
        if yad --question --title="$progname" --text="Открыть видео в mpv?" --width=$cvdw --
text-align=center --buttons-layout=center --center --button='Да!answer:0' --
button='Нет!download:1' ;
then mpv=1;fi
        if [ "$temp" = '480p|' ];
then temp=2;fi
        if [ "$temp" = '720p|' ];
 then temp=3;fi
        if [ "$temp" = '1080p|' ] ;
 then temp=4;fi
        if [ "$temp" = '360p|' ];
 then temp=1;fi
        fi
        elif [ "$listing" = " ];
then
        if [ $yad = 1 ]; then yad --text='Ничего не найдено..' --buttons-layout=center --center --
button='Блин блинский!face-cool';else echo Ничего не найдено..
        fi
        exit
        fi
```

```
        if [ $yad = 0 ];
 then read temp
        temp=$(echo $temp) | sed 's/^[ \t]*//;s/[ \t]*$//'
        fi
        if (( $debug > 1 ));
then echo seems temp=$temp, all wahat i know; fi
        if [[ "$temp" = *mpv ]];
then mpv=1
        elif [ -z $mpv ];
 then mpv=0
        fi # проверка актуальна для gui
        if [ "$listing" = anilibria ];
then
        if [ "$lfhd" != 0 ] && [[ "$temp" = 3* ]];
 then res=-hd;
err=no;
video=$lfhd
        elif [[ "$temp" = 2* ]];
 then err=no
        elif [[ "$temp" = 1* ]];
 then res=-sd;err=no
        else echo Кринжанул, брат;
err=yes;fi
        elif [ "$listing" = jutsu ];
then
        if [[ "$temp" = 2* ]];
then
err=no;
video=$notso
        elif [[ "$temp" = 1* ]];
 then
err=no;
video=$sd
        elif [ "$hd" != 0 ] && [[ "$temp" = 3* ]];
 then
```

```
err=no;
video=$hd
        elif [ "$lfhd" != 0 ] && [[ "$temp" = 4* ]];
 then
err=no;
video=$lfhd
        else echo Кринжанул, брат;
err=yes
        fi;fi
        if [ "$listing" = jutsu ];
then ffprobe -v error -select_streams v:0 -show_entries stream=width,height -of csv=s=x:p=0
$video -user_agent " > .cvd.reso.tmp &
        fi
        if (("${#num}" < "4")) && [ "$listing" = anilibria ];
then
num=0001;fi
        if [ "$err" = no ];
then
        if [ "$listing" = anilibria ];
then    if (( $debug > 1 )); then echo "hey, im trying to ffprobe -v error -select_streams v:0 -
show_entries stream=width,height -of csv=s=x:p=0
https://"$ll"videos/ts/$id/$num$res/playlist.m3u8"; fi
        ffprobe -v error -select_streams v:0 -show_entries stream=width,height -of csv=s=x:p=0
https://"$ll"videos/ts/$id/$num$res/playlist.m3u8  | cut -d'
' -f1 > .cvd.reso.tmp &
        fi # получатель разрешения
        fi
        if [[ ~ = /c/Users/* ]];
then
attrib +S +H .cvd.res.tmp > /dev/null
        fi
        done
        if [[ $first != $last ]] && [ $listing = anilibria ] ;
 then
        printf '0- весь сезон не 0- выбранная серия'
```

```
        if [ "$last" != "$allnum" ] && (echo "$allnum" | grep -E -q "^?[0-9]+$");
 then
        if (( debug > 1 ));
then
echo "$last";fi
        if (( debug > 1 ));
then
 echo "$allnum";fi
        let 'down = last + 1'
        echo " $down- ждать серию $down и сразу скачать"
        else
echo "
        fi
        if [ $yad = 0 ];
 then
        read temp
        temp=$(echo $temp) | sed 's/^[ \t]*//;s/[ \t]*$//'
        else
        w=$(( ${#names} * 11 + 170 ))
        temp=$(yad --scale --title="$names" --max-value=$last --text='Выбор серии' --min-
value=$first --value=$first --text-align=center --center --width=$w --buttons-layout=center)
        fi

        if (echo "$allnum" | grep -E -q "^?[0-9]+$");
then :
                                else
allnum=$last
                                                fi
        elif [ $listing = anilibria ];
then
 num=$first; fi
        if [ $last = $first ];
then
temp=$first;fi
        if [ "$temp" = 0 ] && [ $mpv = 0 ] && [ "$wtf" != 1 ];
```

```
then #
            temp=$first
            echo $temp
            until [ "$int" = int ]
             do
             num=$(echo 000$temp | cut -c${#temp}-7)
             trap "int=int;
 echo Прерванно." INT
             echo Скачивание серии "$temp"/$allnum
             ffmpeg -i https://"$ll"videos/ts/$id/$num$res/playlist.m3u8 $argsV $num.$cont
2>/dev/null
             if [ $temp = $allnum ];
then break
             fi
             let 'temp = temp + 1'
             done
            if [ "$int" = int ];
then
 rm -rf $num.$cont; fi
            fi
            if [ "$temp" = "$down" ] && [ $mpv = 0 ];
            then
            num=000$temp;
num=$(echo $num | cut -c${#temp}-7)
            temp=0
           until (( $temp < -1 ));
 do
                    trap "temp=-3;
 echo Прерванно." INT
                    let 'temp = temp + 1 '
                    printf "Попытка №$temp, "
                    echo $(date --rfc-3339=seconds | cut -d+ -f1)
                    if [[ $(curl https://"$ll"videos/ts/$id/$num$res/playlist.m3u8 2>> .cvd.curl.tmp) =
*#EXTM3U* ]] ;
then
```

```
                echo Дождались!
                temp=-2
                elif [ $temp != -3 ];t
hen
sleep $downtime
                fi
                if [[ $(cat .cvd.curl.tmp ) = *curl*6*resolve ]];
then
                echo 'Ошибка curl (6), проверьте доступ в интернет.'
                exit
                fi
        done
        fi
        if [ $listing = anilibria ];
 then
        if [ $temp = -2 ];
then
ffmpeg -i https://"$ll"videos/ts/$id/$num$res/playlist.m3u8 $argsV $num.$cont;fi
        if (( $debug > 1 ));
 then
echo fiecho target 3 done;fi
        if [ $mpv = 0 ];
then
         num=000$temp;
num=$(echo $num | cut -c${#temp}-7)
        # echo
        until [ -z $(cat .cvd.reso.tmp) ]
        do
        echo test
        sleep 1
        done
         ffmpeg -i https://"$ll"videos/ts/$id/$num$res/playlist.m3u8 $argsV "$num"["$(cat
.cvd.reso.tmp)"].$cont #2>&1 | grep -i --line-buffered frame | cut -d= -f2 | sed -u s/fps/"/g
          if (( $debug > 1 ));
 then echo fiecho target 4 done;
```

```
fi;fi
        if [ $mpv = 1 ];
then
        realnum=000$temp;num=$(echo $num | cut -c${#temp}-7)
        if (( debug > 1 ));
then echo seems mpv=1, num=$num, temp=$temp; fi
        until [ $temp = $(($last + 1)) ]
        do
        let 'temp = temp + 1'
        videos="$videos https://"$ll"videos/ts/$id/$num$res/playlist.m3u8"
        num=000$temp;num=$(echo $num | cut -c${#temp}-7)
        done
        if [ -z $(cat .cvd.reso.tmp) ];
 then :;
else resget=', '
        fi
        if [ $first != $last ];
 then
ser=', серия '
        else
        temp="
        fi #videos $mpvargs --playlist-start=$(($realnum)) --force-media-title="$(echo
$names$ser$temp$resget$(cat .cvd.reso.tmp))"
         mpv $videos $mpvargs --playlist-start=$(($realnum - 1 )) --force-media-title="$(echo
$names$resget$(cat .cvd.reso.tmp))"
        fi
        elif [ $listing = jutsu ] && [ $mpv = 1 ];
 then
        echo $jutres > .cvd.jutres.tmp
        mpv --user-agent=" $video $mpvargs --force-media-title="$(echo $names, серия $num,
$(cat .cvd.reso.tmp))"
        elif [ $listing = jutsu ] && [ $mpv = 0 ] && [ $temp != 0 ]; then #echo щас
cREDACTEDдем..
        until [ "$err" = 0 ]
        do
```

```
	temp=$(yad --file --save --confirm-overwrite='Вы уверены, что хотите перезаписать выбранный файл?')
	if [ -z $temp ];
then
err=1
	else
err=0;fi
	done
	#if [[ $temp != *.mkv ]] || [[ $temp != *.mp4 ]] || [[ $temp != *.avi ]] || [[ $temp != *.$cont ]];
 then
temp=$temp.$cont;fi
	cont=$(echo $video | cut -d? -f1 | rev | cut -d. -f1 | rev)
	if [[ $temp != *.$cont ]]; then temp=$temp.$cont;fi
	curl $video -A " -o "$temp"
	fi
if (( $debug < 1 ));
then
rm -rf .cvd.*tmp; fi
if (( $debug > 0 ));
 then
echo first=$first; fi
if (( $debug > 0 ));
 then
echo last=$last;  fi
if (( $debug > 0 ));
 then
echo nums=$nums;  fi
if (( $debug > 0 ));
then
echo allnum=$allnum;fi
if (( $debug > 0 ));
then
echo lfhd=$lfhd;   fi
if (( $debug > 0 ));
```

```
 then
echo ll=$ll;       fi
if (( $debug > 0 ));
then
echo mpv=$mpv;   fi
if (( $debug > 0 ));
 then pwd;                  fi
# if (( $debug > 0 ));
 then echo reso=$reso;
fi ваша REDACTED устарела
if (( $debug > 0 ));
then
echo link=$link;  fi
if (( $debug > 0 ));
then
echo typehaserr=$typehaserr; fi
if (( $debug > 0 ));
then
echo num=$num; fi
if (( $debug > 0 ));
then
echo names=$names; fi
if (( $debug > 0 ));
then
echo err=$err; fi
if (( $debug > 0 ));
then
echo findby=$findby; fi
if (( $debug > 0 ));
then
echo shitnum=$shitnum; fi
if (( $debug > 0 ));
 then
 echo temp=$temp; fi
if (( $debug > 0 ));
```

```
then echo hd=$hd; fi
if (( $debug > 0 ));
then
echo sd="$sd"; fi
if (( $debug > 0 ));
 then e
cho notso=$notso; fi
if (( $debug > 0 ));
then
echo res="$res"; fi
if (( $debug > 0 ));
then
echo jutres=$jutres; fi
if (( $debug > 0 ));
 then
echo video=$video; fi
if (( $debug > 0 ));
 then
echo listing=$listing; fi
if (( $debug > 0 ));
 then
echo code=$code; fi
if (( $debug > 0 ));
then
echo codeplus=$codeplus;fi
if (( $debug > -1 ));
 then e
cho Затрачено $(( $(date +%s) - $start )) секунд.;fi
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи