

УДК 514.18

Міскевич О. І., Кокоша М. С.

Луцький національний технічний університет

## ДОДАТОК «СИСТЕМНИЙ МОНІТОР» ЗАСОБАМИ БІБЛІОТЕКИ QT.

**Міскевич О. І., Кокоша М. С. Додаток «Системний монітор» засобами бібліотеки Qt.** Розроблено додаток для моніторингу системних ресурсів (обсяг спожитої пам'яті, файлу підкачки, завантаженість процесора, примонтовані файлові системи і т.п.) засобами Qt під ОС Linux. Дана програма може служити для користувачів невеликою утилітою для відслідковування стану їх системи.

**Ключові слова:** Qt, Linux, відкрите програмне забезпечення, системний монітор, диспетчер задач, системне програмне забезпечення

**Міскевич О. И., Кокоша М. С. Приложение «Системный монитор» средствами библиотеки Qt.** Разработано приложение для мониторинга системных ресурсов (объем потребленной памяти, файла подкачки, загрузка процессора, смонтированные файловые системы и т.п.) средствами Qt под ОС Linux. Данная программа может служить для пользователей небольшой утилитой для отслеживания состояния их системы.

**Ключевые слова:** Qt, Linux, открытое программное обеспечение, системный монитор, диспетчер задач, системное программное обеспечение.

**Miskevych O. I., Kokosha M.S. Application «System Monitor» base on Qt Library.** Developed application for monitoring of system resources (like as amount of consumed memory, swap memory, mounted file systems, etc) base on Qt library for Linux OS. This application is a small utility for users that want to monitor performance state of own system.

**Keywords:** Qt, Linux, open-source software, system monitor, task manager, system software.

**Вступ.** Продуктивність – це один з найбільш помітних параметрів системи. Із збільшенням кількості ресурсоємного програмного забезпечення часто постає проблема у її нестачі. Для відслідковування стану системи користувачами чи адміністраторами призначені утиліти і програмне забезпечення, що служать для визначення характеристик заліза комп'ютера, його тестування і моніторингу стану основних компонентів в реальному часі. У даній статті здійснюється короткий огляд існуючих засобів для моніторингу системи, можливостей бібліотеки Qt; операційної системи Linux, а також приділяється увага розробці додатку "Системний монітор".

**Постановка проблеми.** У ході роботи було поставлено завдання створити додаток для моніторингу стану основних параметрів системи, що працюватиме під ОС Linux. Для розробки додатка використано мову програмування C++ та середовище розробки Qt Creator.

**Аналіз існуючих рішень.** Більшість дистрибутивів Linux поставляється з великим числом програмного забезпечення та утиліт, які призначені для моніторингу системних ресурсів і продуктивності.

До цього числа можна включити такі основні консольні інструменти:

- top (консольна команда, що виводить список виконуваних процесів і основну інформацію про них), а також деякі її модифікації – atop, htop.
- iotop – утиліта, подібна до утиліти top, але відображає використання не CPU і пам'яті, а роботу процесів з дисками, написана на Python. Допоможе визначити який процес звертається до жорсткого диска в Linux. Відображає активні процеси, які в даний момент виконують операції запису / читання з диску, збирає статистику за певний час.
- powertop – утиліта, що дозволяє виявити в системі компоненти, які споживають більше енергії, ніж потрібно і показує загальне електроспоживання (у Вт), інформація зчитується з різних джерел ядра. Це дозволяє експериментувати з налаштуваннями для керування електроживленням, ефективно налаштувати споживання енергії під вашу машину.
- iftop – виводить інформацію про активні мережеві з'єднання, швидкість мережевого завантаження / віддачі, моніторить онлайн-трафік, розділяє трафік за протоколами, інтерфейсами і хостами.
- vnStat – утиліта для обліку мережевого трафіку, зберігає історію мережевого трафіку для обраних інтерфейсів. vnStat отримує дані з ядра Linux.

А також графічні інструменти:

- Gnome system monitor – утиліта виводить у вигляді графіків інформацію в реальному часі про ресурси - використання процесора (CPU), використання оперативної пам'яті (RAM) і файлу підкачки (SWAP), а також використання мережі.
- KDE System Guard – менеджер завдань та монітор ресурсів і продуктивності, що є частиною стільничного середовища KDE. Фактично має ті самі функції, що gnome system monitor. Написано на C++ (Qt).
- Psumon – зручний графічний додаток для перегляду поточних процесів і моніторингу стану системи. Написано мовою Python (Qt). Psumon має простий користувальницький інтерфейс і мінімум налаштувань, дозволяє в режимі реального часу стежити за використанням системних ресурсів і системними процесами з можливістю їх примусового завершення.
- Conky – вільна програма для моніторингу стану системи для X Window System. Характерною ознакою Conky є надзвичайна гнучкість в налаштуваннях, що дозволяє кожному користувачеві підлаштувати її інтерфейс та вивід системної інформації за своїми уподобаннями. На відміну від інших програм для системного моніторингу, які використовують спеціальні складні віджети, Conky відображається напряму у вікні X Window. Це дозволяє використовувати менше системних ресурсів у порівнянні з іншими програмами.

Це далеко не всі програми для моніторингу системних ресурсів, представлені у світі open-source, їх набагато більше. Для наглядності на рис. 1 зображено open-source інструментарії та бібліотеки для розробки серверного, мережевого ПЗ та програмного забезпечення із графічним інтерфейсом користувача.

Як бачимо з даної схеми Qt займає чільне місце серед основних інструментаріїв для розробки ПЗ з графічним інтерфейсом користувача. Ще одним із таких фреймворків є GTK+ - кросплатформовий набір інструментів для створення графічних інтерфейсів користувача. Разом із Qt є одним із найпопулярніших інструментів для X Window System. Gnome system monitor (GSM), згаданий вище, написаний за допомогою цього інструментарію.

У власному додатку було використано середовище Qt. Додаток за своїм функціоналом є облегшеною версією Gnome system monitor.

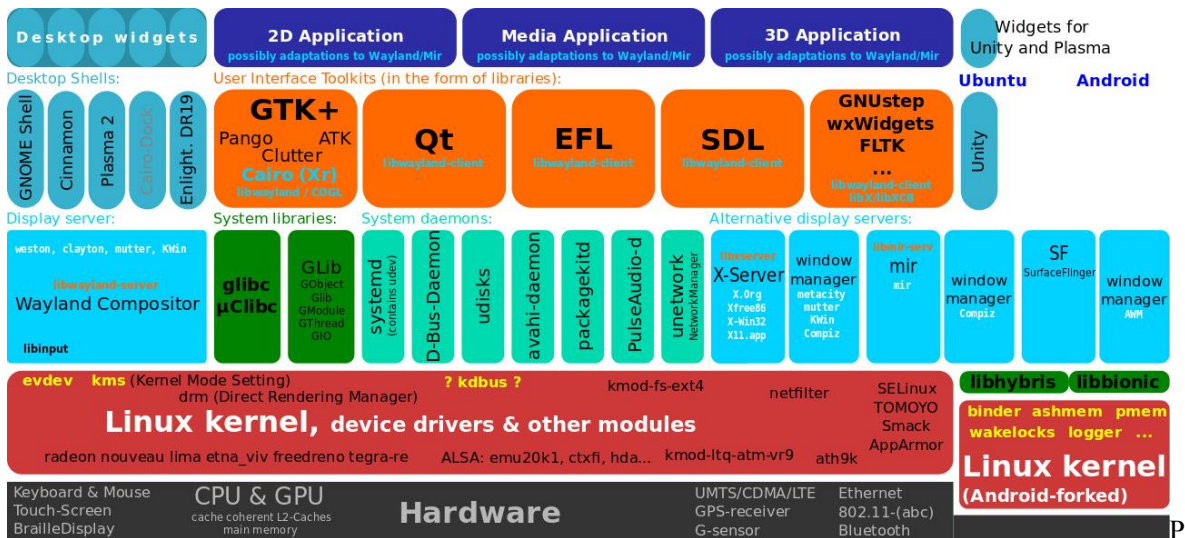


рисунок 1 - Open-source інструментарії для розробки ПЗ

**Виклад основного матеріалу роботи.** Робота додатку "Системний монітор" по-суті базується на парсингу деяких підкаталогів віртуальної файлової системи *proc* та використанні стандартних бібліотек та засобів C++ і бібліотеки Qt. Для кращого розуміння необхідно познайомитися з */proc* ближче.

Файлова система */proc* є механізмом для ядра і його модулів, що дозволяє надсилати інформацію процесам (звідси і назва */proc*). За допомогою цієї віртуальної файлової системи можна працювати з внутрішніми структурами ядра, отримувати корисну інформацію про процеси і змінювати конфігурацію (змінюючи параметри ядра) на льоту. За допомогою файлів в */proc* можна отримати інформацію про стан ядра, процесів, параметрах комп'ютера і т.д. Більшість файлів в */proc* містять найсвіжішу інформацію про системне обладнання. Незважаючи на те, що ці файли віртуальні - їх можна переглянути будь-яким текстовим редактором або за допомогою команд "more", "less" або "cat". [1] Перелічимо деякі важливі файли:

- */proc/cpuinfo* - інформація про процесор (модель, сімейство, розмір кеша і т.д.).
- */proc/meminfo* - інформація про RAM, розмір файлу підкачки і т.д.
- */proc/mounts* - список змонтованих файлових систем.
- */proc/devices* - список пристроїв.
- */proc/filesystems* - підтримувані файлові системи.
- */proc/modules* - список завантажуваних модулів.
- */proc/version* - версія ядра.
- */proc/cmdline* - список параметрів, переданих ядру при завантаженні. [2]

Насправді файлів в каталозі */proc* набагато більше. Можна детально переглянути всі файли в */proc* командами "more" і read. Для прикладу наведу функцію, що отримує загальний процесорний час із каталогу */proc/stat*:

```
unsigned long long getTotalCpuTime()
{
    FILE* file = fopen("/proc/stat", "r");
    if (file == NULL) {
        perror("Неможливо зчитати файл");
        return 0;
    }
    char buffer[1024];
    unsigned long long user = 0, nice = 0, system = 0, idle = 0;
    unsigned long long iowait = 0, irq = 0, softirq = 0, steal = 0, guest = 0, guestnice = 0;
    char* ret = fgets(buffer, sizeof(buffer) - 1, file);
    if (ret == NULL) {
        perror("Неможливо зчитати файл");
        fclose(file);
        return 0;
    }
    fclose(file);
    sscanf(buffer,
           "cpu %16llu %16llu %16llu %16llu %16llu %16llu %16llu %16llu %16llu %16llu",
           &user, &nice, &system, &idle, &iowait, &irq, &softirq, &steal, &guest, &guestnice);
    // http://unix.stackexchange.com/q/178045/20626
    return user + nice + system + idle + iowait + irq + softirq + steal;
}
```

Так як для розробки було вибрано бібліотеку Qt, то давайте дещо ознайомимося з нею.

Qt – багатоплатформовий інструментарій розробки ПЗ на мові програмування C++. Є також «прив'язки» до багатьох інших мов програмування: Python - PyQt; Ruby - QtRuby; Java - Qt Jambi та інші.

З часу своєї появи в 1996 році бібліотека Qt лягла в основу тисяч успішних проєктів по всьому світу. Крім того, Qt є фундаментом популярного робочого середовища KDE, що входить до складу багатьох дистрибутивів Linux.

Відмітна особливість Qt від інших бібліотек — використання Meta Object Compiler (MOC) — попередньої системи обробки вихідного коду. MOC дозволяє у багато разів збільшити потужність бібліотек, вводячи такі поняття, як слоти і сигнали. Утиліта MOC шукає в заголовних файлах на C++ опису класів, що містять макрос Q\_OBJECT, і створює додатковий вихідний файл на C++ , що містить метаоб'єктний код [3].

Перелічимо основні переваги Qt над іншими інструментаріями:

- Використання сигналів і слотів замість функцій зворотнього виклику, що значно спрощує розуміння коду і робить його лаконічним
- Qt Assistant — детальна довідкова система Qt, що спрощує роботу з документацією по бібліотеці.
- Відсутність потреби слідкувати вручну за виділенням пам'яті — у Qt при видаленні класу-предка усі класи-нащадки знищуються автоматично.
- Можливість написання кросплатформенного коду — треба лише скопіювати код під платформу, на якій працюватиме програма.

Отже, перейдемо до UML діаграми класів додатку (рис. 3). Вона показує структуру класів програми — її атрибути і методи, а також взаємозв'язки між об'єктами самих класів.

Пройдемо по призначенню основних класів програми.

Класи `ProcessInformationWorker`, `resourcesWorker`, `fileSystemWorker` — фактично є каркасами трьох вкладок додатку — процеси, пам'ять і файлові системи (рис. 2)

Класу `processPropertiesDialogue` відповідає діалогове вікно, що містить таблицю із більш детальними властивостями вибраного процесу.

Клас `workerThread` відповідає за управління потоками для кожної із 3-х вкладок додатку і діалогових вікон.

Клас `MainWindow` відповідає за будову головного вікна програми, старт і своєчасне видалення класів потоків при виході з програми чи при раптовому її зупиненні, створення головного меню та виклик діалогового вікна.

Класи `TableMemoryItem` та `TableNumberItem` відповідають за коректне порівняння значень колонок у таблицях.

Класи `QObject`, `QMainWindow`, `QTableWidgetItem`, `QTableWidget`, `QDialog`, `QAction`, `QCheckBox` та інші — рідні класи бібліотеки Qt, призначені для програмування графічного інтерфейсу користувача (детальніше про їх методи та властивості можна дізнатися у довідковій системі Qt Assistant).

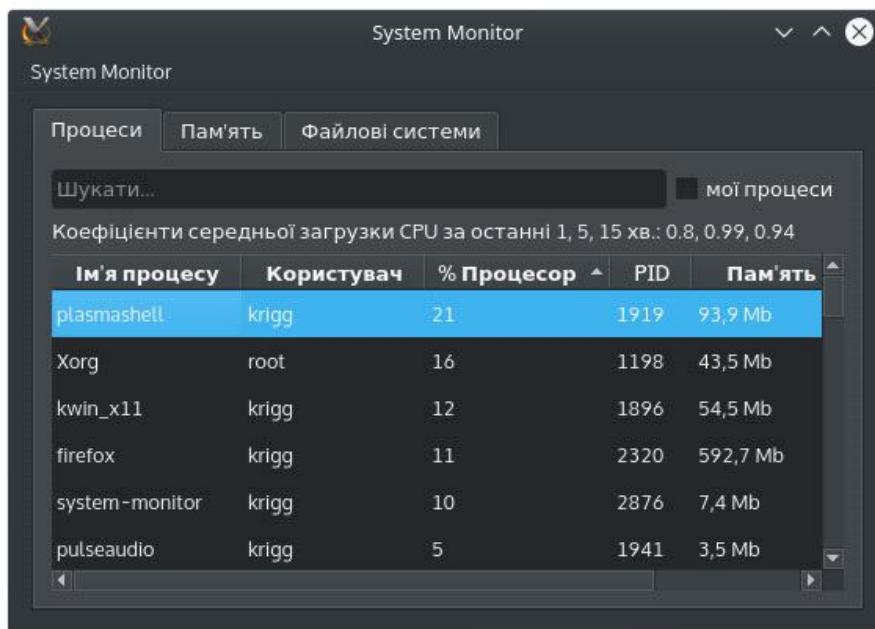


Рисунок 2 - Загальний вигляд головного вікна програми

