**Лекція 12.**

Поки що Windows — найпопулярніша операційна система для ПК, Linux найчастіше вибирають для вебсерверів. Навіть Microsoft використовує Linux для своїх хмарних середовищ. У статті розповідаємо про найкращі дистрибутиви для серверів та для яких завдань вони підходять.

Linux Kernel — це ядро операційної системи, яке спеціалісти використовують для побудови дистрибутивів Linux. Воно об’єднує тисячі інженерів у всьому світі і є ядром не лише для комп’ютерів, жорстких дисків і моніторів, а й для багатьох Linux-ком’юніті — звісно, у переносному сенсі.

Linux Kernel, або ядро Лінукс управляє ресурсами системи (часовими інтервалами CPU, що видається процесам, розподілом пам’яті, доступом до жорсткого диска та мережі), дає можливість процесам обмінюватись інформацією. На відміну від Microsoft Windows та MacOS вихідні коди Linux доступні кожному для вільної модифікації та налаштування під будь-які задачі.

Сьогодні Linux вбудована в настільні комп’ютери, смартфони, планшетні ПК, маршрутизатори комп’ютерних мереж (роутери), пристрої автоматики, системи керування телевізорами та ігрові консолі. Загалом ця операційна система входить у трійку найпопулярніших у світі.

Linux відрізняється від інших операційних систем своєю відкритістю, гнучкими налаштуваннями та стабільною роботою. Ми, як інженери, часто маємо справу з найновішими процесорами, мікросхемами, які з’являться на ринку лише через декілька років, і швидко запустити їх розробникам допомагає саме Linux — можливість вільно додавати підтримку нового апаратного забезпечення до операційної системи та потужне ком’юніті дозволяє працювати з передовими технологіями та швидко створювати нові розробки.

До речі, ОС Android базується саме на ядрі Linux, тому всі, хто використовують Android-пристрої, певним чином мають відношення й до Linux.

Все частіше Linux використовують і на [суперком](https://uk.wikipedia.org/wiki/%D0%A1%D1%83%D0%BF%D0%B5%D1%80%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80)п’юте[рах](https://uk.wikipedia.org/wiki/%D0%A1%D1%83%D0%BF%D0%B5%D1%80%D0%BA%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80) (клас найпотужніших комп’ютерних систем на сьогодні). Цікаво, що станом на 2019 рік усі 500 найпотужніших комп’ютерів світу використовували Linux як свою операційну систему.

### Встановити Linux може будь-хто на (майже) будь-що

Операційною системою Linux активно користуються професійні програмісти, програмісти-аматори, користувачі, яким подобається можливість налаштовувати операційну систему під себе, конфігурувати та підганяти її; а також ентузіасти, що встановлюють її на самі неочікувані пристрої — холодильники, паяльники, автомобілі чи смартфони.

Також Linux з різних причин користуються ті, кому не подобається Windows чи MacOs. Windows — власницька ліцензована система. Це означає, що в правах використання юзери обмежені тим, що їм дозволяє компанія-виробник цієї ОС. На противагу цьому Linux має повністю відкрите ядро ​​ОС та код програм. Таким чином, його можна використовувати у будь-яких цілях абсолютно безплатно та налаштовувати й переробляти під свої потреби.

### Гнучкі налаштування та стабільність — переваги Linux для фахівця

**Відкритість і доступність** — головна ідеологія Linux. Цю операційну систему можна легко встановити ноутбук або ПК, вона безплатна, має багато дистрибутивів — варіантів Linux на будь-який смак, які готові для кінцевої установки на машину. Вони бувають і комерційними (наприклад, RHEL), однак платити потрібно лише за оновлення і технічну підтримку.

Крім того, маючи хороші скіли в програмуванні, можна самостійно змінювати код програм. Linux та більшість софту його екосистеми мають відкриті ліцензії, а тому вихідні коди програми можна переглянути на публічних серверах чи завантажити. Децентралізована мережа фахівців з усього світу переглядає і рецензує код перед додаванням в Linux на предмет помилок та оперативно їх виправляє. Це «самоочищує» ОС від багів.

Також серед переваг Linux — **безпека та децентралізація**. Ця ОС добре зарекомендувала себе як достатньо захищена від зламів, а значить, від вірусів, майнерів і рекламних модулів. Її відкриті програми належать одночасно всім і нікому конкретно.

Різні дистрибутиви, які створюють користувачі, часто суттєво відрізняються один від одного. Наприклад, одні дистрибутиви фокусуються на корпоративній екосистемі та підтримці (наприклад, RHEL, Ubuntu, Calculate), інші — на універсальності та потужності (наприклад, Debian, Arch). Також є вузькоспеціалізовані варіанти дистрибутивів Linux (як для офісних працівників чи звукових студій) або ті, які підходять для кастомізації систем та прикладного софту, наприклад Gentoo, Yocto, BuildRoot.

В ОС **немає обмежень** на кількість процесорів, ядер, обсяг оперативної пам’яті, а тому Linux може працювати як на мінімалістичних вбудованих системах типу наручних годинників, так і на суперкомп’ютерах. Настільний Linux простіший у використанні, ніж, наприклад, Windows. Все, що потрібно робити — це встановити/ видалити програму та оновити систему через менеджер пакетів, причому ОС майже не накопичує помилок, хоча кількість встановлених/ видалених пакетів може обчислюватися тисячами.

Однією з переваг Linux є також її**власне ком’юніті.**  Оскільки Linux поширений саме серед розробників, інженери постійно покращують софт і очищають його від різноманітних «багів».

### **Слабкі місця Linux**

Linux сильно відрізняється від Windows, з яким всі звикли працювати, а тому в користувачів часто виникає **проблема в освоєнні цієї операційної системи**. Звичайно, якщо користуватись ОС на рівні «лайкати котиків» на Facebook чи Instagram або позависати на YouTube, то різниця невелика. Але Linux має потужний інструмент у вигляді інтерфейсу командного рядка зі своєю ідеологією. Звикати до нової концепції ОС завжди складно, однак реально. У розробників, наприклад, не виникає особливих проблем з адаптацією до ОС, а з часом ефективніша робота з вбудованими інструментами Linux захоплює, і відмовитись від них вже не вийде :)

Ще одне слабке місце Linux — **відсутність деяких версій популярних програм та інколи хороших драйверів для периферії**. Наприклад, для цієї ОС немає Microsoft Office та Photoshop. Також відсутні інші специфічні для деяких професій програми. Однак для багатьох з них є аналоги. Якби всі необхідні програми були адаптовані під Linux, то він точно мав би набагато більше користувачів, ніж зараз. З точки зору ж розробника, всі необхідні інструменти були адаптовані під Linux і доступні для легкої установки та використання.

Також не всі виробники периферійних пристроїв випускають хороші драйвери під Linux, таке трапляється з відеокартами чи WiFi-адаптерами, а їх налаштування може зайняти деякий час, і, як бонус, дає можливість поглиблено розібратись з нутрощами Linux.

Інколи підтримки якогось пристрою може не бути взагалі в Linux, і тоді працює правило — написати драйвер самому, чи заплатити іншому Linux Kernel інженеру, щоб він його написав, або просто чекати, доки хтось в Linux ком’юніті його не створить.

## Linux чи MacOS: яку операційну систему обрати

Це провокаційне питання, адже програмування під Linux і під MacOS часто використовується для вирішення різного типу задач. Але для порівняння: операційна система Linux схожа на MacOS, проте має іншу історію. MacOS встановлюється на обмеженому наборі пристроїв і постачається разом із комп’ютерами компанії Apple. Linux, навпаки, встановлюється на все, навіть на ті пристрої, на яких вже стоїть MacOS.

MacOS — це операційна система сімейства Unix. Простіше кажучи, вона як Linux (яка теж Unix-подібна ОС), проте із закритим вихідним кодом. Тому, якщо ви раніше програмували в MacOS чи користувались командним рядком, у Linux ви побачите схожий функціонал та використання стандартів POSIX.

## То чим займаються Linux Kernel інженери

Linux Kernel інженер — це людина, яка розробляє рішення на основі Linux. Основне в цій професії — базові знання і розуміння специфіки розробки для ядра ОС. Linux — це та операційна система, з якої найчастіше починають виробники нових процесорів, інженери часто починають працювати із залізом, яке з’явиться на ринку лише через декілька років. В таких напрямках потрібне розуміння цифрової електроніки, внутрішньої будови процесора, кешу, пам’яті, а задача інженера — запустити Linux на новому процесорі чи системі-на-чипі, тож паяльник чи осцилограф часто зустрічаються на їхньому робочому столі.

Також Linux Kernel інженери додають нові фічі-родзинки до операційної системи. Наприклад, створюють драйвери для того, щоб користувач міг під’єднуватися до WiFi-точок нового покоління. Або створюють драйвери для нових файлових систем.

Дуже багато роботи з мережевим стеком — більшість пристроїв з Linux підключаються до Internet, а з розвитком Internet of Things їх значно побільшало. В системах, критичних до споживання енергії, наприклад, смартфонах та автомобільних бортових комп’ютерах, Linux Kernel фахівці займаються оптимізацією роботи Linux та ‘заліза’ для мінімізації витрат енергії. Ну а задачі Linux Kernel інженерів, що працюють з Android, дуже широкі — і робота з аудіо та мультимедіа, і з USB та зарядкою, і з тачскрінами та дисплеями, і з різноманітними датчиками та бездротовими рішеннями як NFC, BlueTooth та WiFi.

Сетап, на якому були написані драйвери для USB Audio Class 3 для Linux

Розробники Linux та Linux Kernel часто багато працюють із «залізом» (процесори, мікросхеми, і т. д). І бувають випадки, коли розробники «заліза» допускають помилки, а програмістам доводиться розгадувати нетривіальні загадки та дійсно бути хакерами, та зрозуміти, яким чином змусити його працювати.

**1. Ubuntu Server**

Ubuntu Server від Canonical — дистрибутив і для ПК, і для серверів. Саме він посідає перші місця всіх рейтингів. Хоча його часто вибирають для програм, Ubuntu також дозволяє створювати хмарні платформи. Для цього необхідно завантажити Ubuntu Cloud Server.

За даними Canonical, на Ubuntu працює понад 55% хмар OpenStack, які часто використовують для створення інфраструктурних хмарних сервісів та хмарних сховищ.

**Характеристики Ubuntu Server:**

* Безплатний, простий у встановленні дистрибутив зі стандартним циклом випуску;
* Доступний у всіх хостинг-провайдерів Linux;
* За замовчуванням включає популярну VPN WireGuard;
* Використовує менеджер пакетів apt;
* У дистро велика підтримка спільноти, а у LTS-версії — 10 років підтримки та оновлень з боку розробників;
* Сумісний з більшістю додатків та ПЗ.

**Мінімальні вимоги:**

* Оперативна пам’ять: 1 Гб
* Жорсткий диск: 2,5 Гб
* Процесор: 1 ГГц

**2. Debian Server**

Це один із перших дистрибутивів Linux. На ньому засновані багато дистрибутивів, наприклад, Ubuntu та RHEL. Особливість Debian у її стабільності: вона важлива для сервера, тому перекриває всі недоліки дистрибутива. Ця стабільність допомагає уникнути ситуації, коли оновлення можуть конфліктувати з наявним ПЗ.

Debian доступний у кількох варіантах: стабільному, тестовому та нестабільному. У стабільної версії — стандартний цикл випуску з більш тривалим терміном підтримки; у версії для тестування використовується ковзний цикл випуску. А нестабільний варіант не стільки випуск, скільки версія дистро в процесі розробки, що включає останні пакети Debian.

Дистро можна встановити через інтернет за допомогою завантажувального образу. Це дозволить створити сервер з нуля. А ще є хмарний образ: його можна розгорнути на різних провайдерах хмар, наприклад, Amazon EC2, Azure, OpenStack та інших.

**Характеристики Debian:**

* Безплатний дистрибутив, доступний у всіх хостинг-провайдерів;
* За замовчуванням використовує менеджер пакетів apt, що означає потужну систему упаковки ПЗ. Поставляється з більш ніж 50 000 пакетів;
* Дистро підтримує багато комп’ютерних архітектур, таких як amd64 та Aarch64. Має 32-бітну версію;
* Надійний, легкий дистрибутив.

**Мінімальні вимоги:**

* Оперативна пам’ять: 512 Mб
* Жорсткий диск: 2 Гб
* Процесор: 1 ГГц

**3. Red Hat Enterprise Linux**

Red Hat Enterprise Linux (RHEL) — найпопулярніший комерційний дистрибутив. Особливість RHEL — одна з найтриваліших програм довгострокової підтримки. Найкраще підходить для хмарних серверів та центрів обробки даних.

Більшість хостинг-провайдерів не дають готового образу ОС для RHEL. Тому вам потрібно скористатися опцією “ІSO користувача” і завантажити RHEL iso самостійно.

**Характеристики RHEL:**

* Платний дистрибутив Linux, але з відкритим кодом;
* Надає комерційну підтримку та 10 років підтримки LTS-випусків;
* Дозволяє виправити ядро ​​без перезавантаження;
* Дистро стабільний та надійний;
* Дає багато нестандартних інструментів та ПЗ.

**Мінімальні вимоги:**

* Оперативна пам’ять: 512 Мб
* Жорсткий диск: 1 Гб
* Процесор: 2 ГГц

**4. Fedora Server**

Fedora — дистрибутив, розроблений спільнотою та заснований на RHEL, з найновішим ПЗ та частими випусками. Fedora підійде, якщо ви хочете дотримуватися середовища yum і віддаєте перевагу новим версіям ПЗ. Допомагає розгорнути сервер і на “голому залізі”, і в хмарі.

Крім Fedora Server, є й інші версії Fedora для серверів та інших пристроїв у мережі:

* Fedora CoreOS — дистрибутив з автоматичним оновленням, орієнтований на контейнери.
* Fedora IoT — дистрибутив для екосистем інтернету речей.

**Характеристики Fedora:**

* Безплатний дистрибутив із частими релізами — по 2 випуски щороку;
* Кожен випуск підтримується 13 місяців;
* Fedora використовує нове ПЗ;
* Існують різні версії на вибір.

**Мінімальні вимоги:**

* Оперативна пам’ять: 128 Мб
* Жорсткий диск: 1 Гб
* Процесор: 2 ГГц

**5. OpenSUSE Leap**

OpenSUSE Leap — стабільний дистро з регулярними випусками щорічно. Використовує старіші версії ПЗ, але саме це і дає йому стабільність. Також його часто вибирають через адаптивність та гнучкість: він доступний не тільки для серверів, але і для настільних комп’ютерів і ноутбуків.

Дистро не має окремого ISO-образу для серверів. Встановити сервер можна в інсталяторі OpenSUSE.

**Характеристики OpenSUSE:**

* Безплатний дистро, доступний для більшості хостинг-провайдерів;
* Пропонує варіанти стабільної чи безперервної версії;
* Дистро із багаторежимною архітектурою;
* Є підтримка з боку спільноти;
* Пропонує різні інструменти.

**Мінімальні вимоги:**

* Оперативна пам’ять: 1 Гб
* Жорсткий диск: 10 Гб
* Процесор: 1,6 ГГц

**6. SUSE Linux Enterprise Server**

SUSE Linux Enterprise Server (SLES) — комерційний серверний дистрибутив Linux, популярний серед компаній. Нові випуски SLES виходять кожні 3-4 роки. Дистрибутив пропонує різні інструменти та комерційну підтримку. На SLES можна легко перейти з OpenSUSE Leap.

**Характеристики SLES:**

* Платний дистрибутив із комерційною підтримкою;
* Пропонує тривалий термін підтримки LTS-релізів;
* Дистро стабільний та надійний.

**Мінімальні вимоги:**

* Оперативна пам’ять: 512 Мб
* Жорсткий диск: 3.5 Гб.
* Процесор: 1 ГГц

**7. AlmaLinux**

AlmaLinux — це безплатний дистрибутив Linux з відкритим вихідним кодом, альтернатива старому CentOS. Дистро орієнтований на довгострокову стабільність та забезпечує організаціям надійну платформу.

**Характеристики AlmaLinux:**

* Безплатний дистрибутив;
* Дозволяє легко мігрувати з CentOS;
* Дистро стабільний та з тривалими періодами підтримки;
* AlmaLinux OS 8 підтримуватиметься як мінімум до 2029 року;
* Доступно більшість хостинг-провайдерів.

**Мінімальні вимоги:**

* Оперативна пам’ять: 1 Гб
* Жорсткий диск: 20 Гб
* Процесор: 1 ГГц

**8. Oracle Linux**

Oracle Linux — дистро, скомпільований з вихідного коду RHEL. Цей дистрибутив сумісний із бібліотекою додатків RHEL. Завдяки потужним інструментам, дистрибутив підходить для керування пам’яттю та організації файлів. Його можна використовувати для віртуальних середовищ та систем даних.

**Характеристики Oracle Linux:**

* Безплатний дистрибутив, але із можливістю платної підтримки;
* Сумісний з RHEL;
* Підходить для компаній різних розмірів.

**Мінімальні вимоги:**

* Оперативна пам’ять: 1 Гб
* Жорсткий диск: 1 Гб
* Процесор: 1 ГГц

**9. Rocky Linux**

Rocky Linux — дистрибутив, заснований на RHEL, що розвивається спільнотою. Альтернатива CentOS. Це стабільна система, що дає змогу легко мігрувати з інших дистрибутивів RHEL.

**Характеристики Rocky Linux:**

* Безплатний дистрибутив із низхідною збіркою, як у старих версіях CentOS;
* Стабільний, зі стандартним циклом випуску;
* Дистро із 10-річним життєвим циклом підтримки;
* Доступний для більшості хостинг-провайдерів.

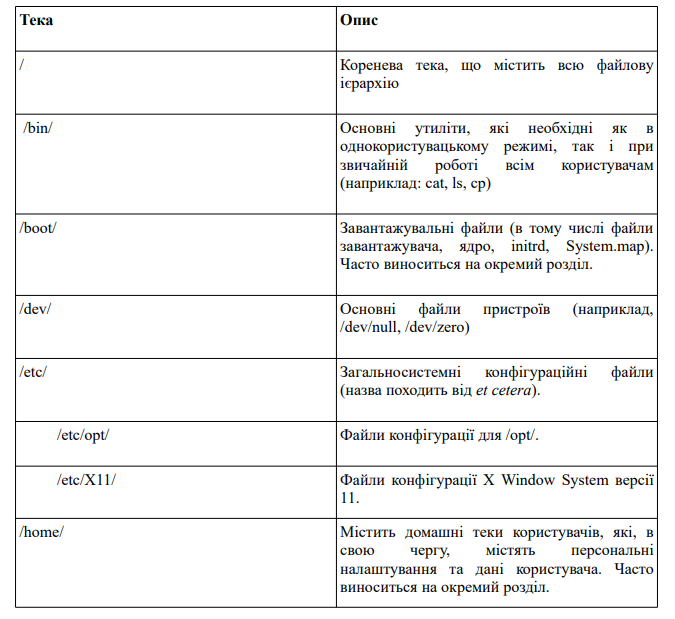
**Мінімальні вимоги:**

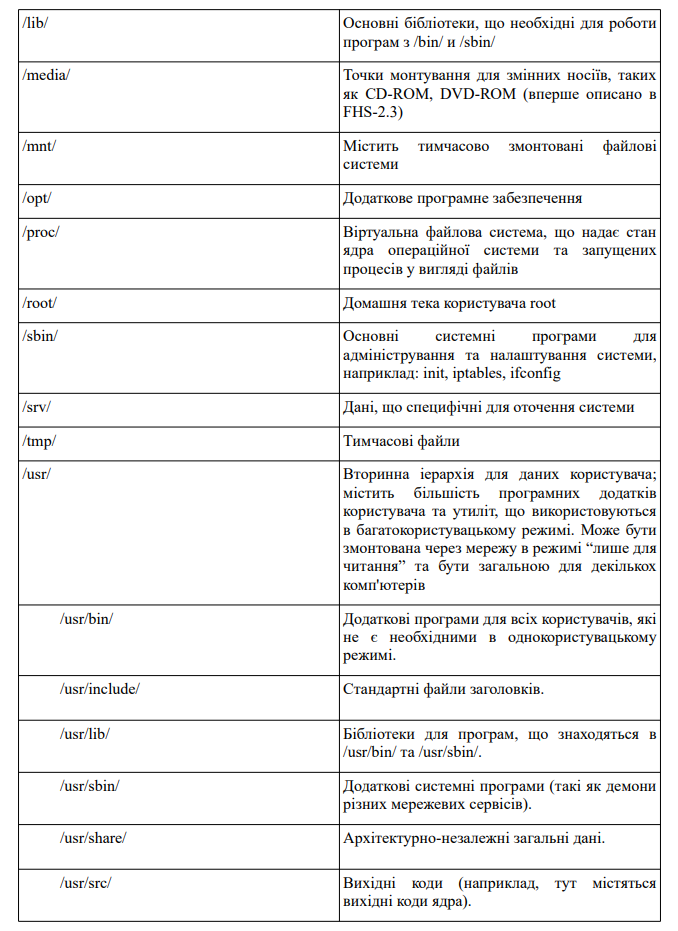
* Оперативна пам’ять: 2 Гб
* Жорсткий диск: 20 ГБ
* Процесор: 1.1 ГГц

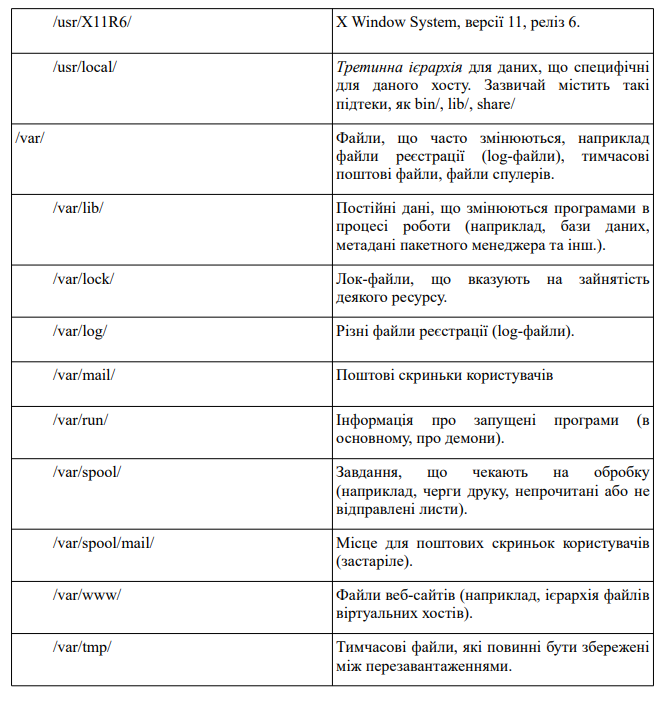
Файлова система Операційна система Linux підтримує велику кількість файлових систем, на сьогодні найбільш широко використовуються: ext2, ext3, ext4, ReiserFS та xfs. Також сучасні ОС Linux сумісні з файловими системами, що використовуються в ОС Windows, такими як NTFS та FAT32, але використання цих файлових систем в Linux небажане, бо ці системи розроблялись під іншу операційну систему і в зв'язку з цим підтримка Windows-розділів в ядрі Linux реалізована за допомогою сторонніх утиліт/драйверів, що накладає деякі обмеження (наприклад, ОС Linux не має можливості розмежовувати права доступу до файлів на розділах NTFS). Структура файлової системи

**Структура файлової системи**

В операційній системі Windows при відкритті теки “Мій комп'ютер” користувач звик до 7 наступної картини: зазвичай один або більше твердих дисків (частіш за все логічних) що називаються починаючи з латинської літери C. Кожен з дисків є кореневою текою. Так, наприклад, якщо в системі є три диска, то буде три кореневих теки (скоріш за все C, D та E), кожен з яких містить вкладені теки та файли. Іншими словами в системі буде існувати три дерева. Оскільки час від часу треба користуватися компакт-дисками та USB-накопичувачами, то періодично будуть “виростати” ще кілька дерев. В дистрибутивах Linux все дещо інакше. Файлова система цілісна і має лише одну кореневу теку, яка позначається косою рискою — слеш (/). Тут треба звернути увагу на відмінність від Windows. В ній при формуванні повної адреси використовується обернена коса риска “\”. В Linux при формуванні повної адреси завжди використовується “/”. Файлова структура Linux має певну структуру тек. В ній все впорядковано та “лежить” на своєму місці. Кожна тека має власне призначення, яке регламентується документом під назвою FHS (Filesystem Hierarchy Standart — стандарт структури файлової системи). Ось короткий опис призначення основних тек згідно даного стандарту







**Користувачі та групи в Linux**

Операційна система Linux є багатокористувацької системою в якій може одночасно існувати досить велика кількість користувачів. В Linux робота з користувачами містить набір наступних маніпуляцій: додавання, видалення або модифікація налаштувань користувача або групи. Крім облікових записів, які використовують люди для роботи з системою, в Linux передбачені облікові записи для системних користувачів: з точки зору системи це такі ж користувачі, однак ці облікові записи використовуються виключно для роботи деяких програм-сервісів. Наприклад, стандартний системний користувач mail використовується програмами доставки пошти. Коли в системі створюється новий користувач, він додається щонайменше в одну групу. В системі при створенні нового облікового запису створюється спеціальна група, назва якої співпадає з іменем нового користувача і користувач включається в цю групу. В подальшому адміністратор може додати користувача і в інші групи. Механізм груп може використовуватись для організовування спільного доступу кількох користувачів до певних ресурсів. Наприклад на сервері компанії для кожного проекту може бути створена окрема група, в яку будуть входити облікові записи співробітників, що працюють над цим проектом. При цьому файли, що відносяться до проекту, можуть належати цій групі та бути доступними для її членів. В системі також існує певна кількість груп (наприклад bin), які використовуються для керування доступом системних програм до різних ресурсів. Як правило членами таких груп є системні користувачі, а користвачі-люди до таких груп не входять. Також за допомогою груп можуть бути надані права, що є необхідними для виконання певних задач. Наприклад для додавання, видалення або налаштування принтера в системі користувач повинен входити в групу lpadmin. Керування користувачами та групами Створення користувача # adduser username Створення групи # addgroup groupname Додавання користувача в групу 1 спосіб # addgroup username groupname 2 спосіб # usermod -aG groupname username Видалення користувача з групи # delgroup username groupname Видалення користувача # deluser username

**Права доступу до файлів та тек**

Оскільки система Linux з самого початку розроблялась як багатокористувацька система, в ній передбачений такий механізм, як права доступу до файлів та тек. Він дозволяє розмежовувати повноваження користувачів, які працюють в системі. Зокрема, права доступу дозволяють окремим користувачам мати «особисті» файли та теки. Правильне налаштування прав доступу дозволяє підвищити надійність системи, захищаючи від змін або видалення важливі системні файли. А оскільки зовнішні пристрої з точки зору Linux також являються об'єктами файлової системи, механізм прав доступу можна застосовувати й для керування доступом до пристроїв. У будь-якого файла у системі є власник — один з користувачів. Однак кожен файл одночасно належить ще й до деякої групи користувачів системи. Кожен користувач може входити в будь-яку кількість груп, і в кожну групу може входити довільна кількість користувачів з числа тих, які визначені в системі. Права доступу визначаються по відношенню до трьох типів дій: читання, запис та виконання. Ці права можуть бути надані трьом класам користувачів: власнику файла (користувачу), групі, якій належить файл, а також всім іншим користувачам, що не входять в цю групу. Право на читання дає користувачу змогу читати вміст файла або, якщо такий доступ дозволений до теки, продивлятися вміст теки (використовуючи команду ls). Право на запис дає можливість користувачу записувати або змінювати файл, а право на запис для теки — можливість створювати або видаляти файли в цій теці. Врешті, право на виконання дозволяє користувачу запускати файл як програму або сценарій командної оболонки (зрозуміло, що ця дія має сенс лише в тому випадку, коли файл є програмою або сценарієм). Володіння правами на виконання для теки дозволяє перейти (командою cd) в цю теку. Візуальне значення прав доступу записується двома способами — в символьному вигляді, або у вигляді вісімкового числа. У символьному вигляді це r, w, x або їх комбінація. У вісімковому відображенні кожному виду доступу відповідає число x відповідає 1, w — 2, а r — 4 Комбінація символьних значень в вісімковому вигляді буде виглядати як сума відповідних значень. Символьний вигляд Вісімковий вигляд r 4 w 2 x 1 Так rwx — тобто повний доступ у вісімковому вигляді буде 1 + 2 + 4 = 7 rw- — доступ на читання та запис — 2 + 4 = 6 а r-x — доступ на читання и виконання — 1 + 4 = 5 Повний запис прав доступу до файлу складається з трьох частин: право власника, право групи та право всіх інших та записується відповідно

rwxr-xr-- — в символьному вигляді або 754 — в вісімковому вигляді

Перегляд встановлених прав доступу: $ ls -l

Зміна прав доступу до файла 1 спосіб — явне вказання всіх прав chmod 640 ~/test.txt

2 спосіб — зміна певного права доступу chmod g+w ~/test.txt

Аналогічним чином можна змінювати і власника файлу # chown newuser:newgroup ~/test.txt

**Атрибути файлів і тек за замовчуванням.**

umask. umask — функція середовища POSIX, що змінює права доступу, які надаються новим файлам та текам за замовчуванням. umask зазвичай встановлюється у вісімковій системі числення. Фактично, umask вказує, які біти треба скинути в правах на файл, що встановлюються — кожен встановлений біт umask забороняє встановлення відповідного біту прав. Для себе слід запам'ятати — якщо ви створюєте файл, то значення маски віднімається від значення 666, а якщо створюєте теку, то значення маски віднімається от значення 777. Продивитися поточне значення umask $ umask За замовчуванням значення umask — 22. Це значить, що всі файли створюються з правами 666 — 22 = 644, тобто rw-r--r--, або, простіше кажучи, у володаря файла є права на читання та запис, але не на виконання, а у інших (як у групи, так і у всіх інших користувачів) права лише на читання. Зміна umask $ umask 002 Тепер при створенні нового файлу він з'явиться з правами 666 - 2 = 664, тобто rw-rw-r—

**Біти атрибутів SUID та SGID**

В Unix-подібних системах програма запускається з правами користувача, який викликав вказаний додаток. Це забезпечує додаткову безпеку, так як процес з правами користувача не зможе отримати доступ на запис до важливих системних файлів, наприклад /etc/passwd, який належить суперкористувачу root. Якщо на виконуваний файл встановлений біт suid, то при виконанні ця програма автоматично змінює «ефективний userID» на ідентифікатор того користувача, який є власником цього файла. Тобто, в незалежності від того — хто запускає цю програму, вона при виконанні має права власника цього файла. Встановлення біту suid на файли, що не є виконувальними, не має сенсу. Також немає рації встановлювати цей біт на теку. В останніх версіях ядра Linux додане ще одне обмеження — з міркувань безпеки suid-біт може застосовуватись лише до бінарних файлів, але аж ні як щодо скриптів. Коли груповий s-бит встановлюється для теки, то кожен файл, що буде створено в цій теці буде віднесено до тієї групи, до якої відноситься і сама тека (але не користувач, що створює цей файл). Використання цього біта має сенс лише відносно теки. Встановлення SGID-біта # chmod g+s ~/testdir Зняття SGID-біта # chmod g-s ~/testdir

**Додаткові атрибути файлів**

Крім права на читання/запис/виконання файлові системи ext2, ext3 та ext4, які використовуються в Linux підтримують додаткові атрибути файлів. Необхідно зразу відзначити, що в більшості випадків вам не прийдеться з ними працювати, але знати про те, що вони існують потрібно. Коротко опишемо деякі з них. A - не оновлювати час доступу до об'єкту. Теоретично встановлення цього атрибуту має підвищити продуктивність файлової системи і, відповідно, системи в цілому. а - вказує, що в файл можна додавати інформацію, але не можна видаляти. d - вказує на те, що не треба робити резервні копії файла. Файл буде проігноровано командою dump. i - вказує на те, що файл не можна видаляти та модифікувати. s - вказує, що при видаленні файла місце де був розміщений файл буде перезаписано нулями. u - вказує на те, що при видаленні файла його треба кудись зберегти. Повний список атрибутів можна побачити прочитавши $man chattr Існують і інші атрибути. Про них ви можете прочитати в довідці. Також треба звернути увагу на те, що не всі атрибути працюють на всіх файлових системах.

Змінювати додаткові атрибути можна командою chattr, а для того, щоб їх продивитися існує команда lsattr.

**Монтування дисків. Mount та fstab.**

Розглянемо процедуру монтування файлових систем. Монтування може здійснюватися автоматично, або вручну. mount — утиліта командного рядка в UNIX-подібних системах. Застосовується для монтування файлових систем.

**Ручне монтування локальних ресурсів.**

Команда mount. Варіант команди монтування, що зустрічається найчастіше, буде виглядати так: mount -t тип\_файлової\_системи -o опції\_монтування що\_монтуємо куди\_монтуємо Наприклад, монтування ntfs-розділу на нашому ж жорсткому диску в теку /mnt/win буде виглядати так: mount -t ntfs-3g -o defaults /dev/sda5 /mnt/win де ntfs-3g — тип файлової системи defaults — всі опції монтування за замовчанням /dev/sda5 — пристрій, який монтується /mnt/win — тека, в яку буде проведено монтування Якщо при монтуванні виникли проблеми з кирилицею на ntfs-розділі, в опціях монтування можна явно вказати кодову сторінку: mount -t ntfs-3g -o locale=uk\_UA.utf8 /dev/sda5 /mnt/win Найчастіше утиліта mount всі потрібні параметри (в тому числі і тип файлової системи) може визначити сама. У такому випадку команда монтування може виглядати так: mount /dev/sdb1 /mnt/new\_disk За допомогою mount можна монтувати не тільки розділи на фізичних пристроях, а й просто файли, наприклад iso-образи дисків. В такому випадку використовується пристрій loop: mount -t iso9660 -o loop /home/user/ubuntu-10.04.3-server-i386.iso /home/ubuntu/

**Автоматичне монтування. Файл fstab.**

Автоматичне монтування файлових систем проводиться за допомогою файлу /etc/fstab. Саме в ньому й описано монтування пристроїв, яке буде здійснено при завантаженні операційної системи. Якщо ми відкриємо цей файл в Ubuntu, то побачимо приблизно таке: proc /proc proc nodev,noexec,nosuid 0 0 UUID=87c97237-0bf0-4a62-bae2-47a850643996 / ext4 errors=remount-ro 0 1 UUID=0af767a3-c740-47fe-87ad-fb495bc1444e /home ext4 defaults 0 2 UUID=45bc8486-ed86-4a86-96db-ea34e4c0b9da none swap sw 0 0

Кожен запис в цьому файлі містить 6 полів. Його формат простий: Пристрій — Точка монтування — Тип файлової системи — Опції монтування — вказування необхідності резервного копіювання — Порядок перевірки розділу. Пристрій може визначатися не тільки через UUID, а й через мітку тома, або просто як файл пристрою, наприклад /dev/sda1. Точку монтування вибираємо самі по необхідності. Вона не обов'язково повинна знаходитися безпосередньо в кореневому розділі. Наприклад у мене на поштовому сервері окремий розділ примонтувати як /var/mail. Тип файлової системи — тут все зрозуміло. В який формат відформатували, те і пишемо. Опції монтування — тут можна дуже багато чого описати. Наприклад: монтувати пристрій тільки для читання, або в режимі читання і запису, кодування імен файлів, чи зберігати час доступу до файлів, логін-пароль для доступу (якщо це мережевий ресурс) і так далі. Ознака необхідності резервного копіювання — зазвичай не використовується і встановлюється в 0. Порядок перевірки розділу (0 — не перевіряти, 1 — встановлюється для кореня, 2 — для решти розділів). Тепер деякі подробиці про опції монтування. Всі опції записуються через кому. async — всі дії введення/виводу будуть проводитися асинхронно noatime — не оновлювати час доступу до файлів defaults — використовувати опції за замовчуванням: rw, suid, dev, exec, auto, nouser і async. exec — дозволяти виконання бінарних файлів noexec — не дозволяти виконання бінарних файлів suid — дозволяти використовувати біти set-user-identifier і set-group-identifier. nosuid — не дозволяти використовувати біти set-user-identifier і set-group-identifier. ro — монтувати пристрій в режимі тільки для читання rw — монтувати пристрій в режимі читання/запису users — дозволяти всім користувачам монтувати і демонтувати цей пристрій Крім цього, варто звернути увагу, що при монтуванні різних файлових систем, опції монтування можуть змінюватися. Особливо це відноситься до мережевих файлових систем, таким як sshfs, curlftpfs, smbfs і їм подібним.

**Система керування пакунками**

Пакет Debian - архівний файл (зібраний утилітою ar), що містить два звичайних архіву \*.tar.gz, один з яких включає скомпільовані виконувальні бінарники (та необхідні для їх роботи компоненти — бібліотеки, конфіги, документацію и так далі, другий же — так звані керуючі файли: контрольні суми, опис залежностей, перед- та післяінсталяційні сценарії. Засоби управління пакетами повинні не тільки забезпечити розгортання архівів і розміщення їх в потрібних місцях файлової системи, але й якимось чином відреагувати на залежності — або просто сповістити про їх порушення, або спробувати задовольнити їх своїми силами. Поняття залежностей в Debian та його похідних відрізняється від прийнятого в більшості інших дистрибутивів (і взагалі ОС Unix-родини). Зазвичай розрізняють лише:

• обов'язкові (або "жорсткі") залежності, без задоволення яких встановлення та робота програми неможлива (наприклад, залежність від системних бібліотек)

• залежності необов'язкові ("м'які"), без задоволення яких програма зберігає працездатність, але їх задоволення додає функціональності (наприклад, залежність links або mc від сервису консольної миші gpm). В Debian залежності мають декілька градацій: обов'язкові (depends), настійливо рекомендовані (recommends), рекомендовані помірно наполегливо (suggests), конфліктуючі (conflicts). Перша градація — це звичайні "жорсткі" залежності. З останніми теж все зрозуміло — це, так сказати, антизалежності. Ну а наполегливо рекомендовані та рекомендовані просто — це два різновиду "м'яких" залежностей. Тобто перша категорія ніби то більш потрібна, ніж Друга. Втім, таке суб'єктивна думка мейнтейнера даного пакета — цілком можливо, що у користувача будуть інші потреби. І це ми врахуємо при виборі засобу керування пакетами. Крім того, специфікою Debian є ще існування так званих перед-залежностей (predepends) — при їх порушенні встановлення пакета навіть не може початися. Втім, з точки зору користувача вони не набагато відрізняються від звичайних залежностей типу depends. Крім залежностей, в системах пакетного менеджменту Debian важливо також поняття пріоритету пакета, що показує ступінь необхідності його для функціювання системи, наприклад: обов'язковий (required), без якого функціювання системи неможливе, основний (base) та важливий (important), також що виявляються практично необхідними, стандартний (standard) — тобто наявний практично в будь повнофункціональної Linux-системи, додатковий (optional) — тут вже ступінь важливості кожен повинен вирішувати для себе сам.

**Засоби керування пакунками**

Що стосується засобів керування пакунками, в Debian та його клонах є великий вибір:

• команда dpkg, призначена для встановлення, конфігурування та видалення одиничних пакунків, але не має власних засобів вирішення залежностей між ними;

• dselect — front-end (оболонка) для dpkg, що працює в текстовому режимі; забезпечує не тільки встановлення/ видалення програм, але й груповий вибір пакунків за цільовим призначенням, а також розв'язок залежностей між ними; • механізм apt — універсальний набір інструментів для керування deb-пакунками, включаючи розв'язок залежностей між ними і навіть побудову з вихідних текстів окремих пакунків та тотальне перезібрання встановленої системи з заданими параметрами компіляції;

• aptitude — оболонка для apt, як за інтерфейсом, так і за функціоналом схожа з dselect;

Команда dpkg розв'язком залежностей не займається, лише повідомлює в відповідних випадках про їх порушення. Інструменти з набору apt обмежуються встановленням необхідних залежностей та виводять список настійливо та помірно рекомендованих пакунків. Користувач може сам визначитись, чи потрібно встановлювати йому ці залежності. Набір apt (Advanced Packaging Tools), як видно з назви — це програмний комплекс, що охоплює всі сторони керування пакунками, аж навіть до їх побудови з вихідних текстів. Він включає в себе майже десяток команд, з яких нині нас зацікавлять лише дві — apt-cache, засіб роботи з кешем пакунків та apt-get - інструмент для їх отримання та встановлення. Конфігураційні файли APT • /etc/apt/sources.list — список джерел пакунків (репозиторіїв) • /etc/apt/apt.conf — основний файл конфігурації APT • /etc/apt/preferences — файл уподобань, керує тим, яка версія пакунка буде встановлена у випадку наявності в репозиторії одразу декількох версій

**Apt-get**

#apt-get update — оновлює список доступних пакунків Якщо ви не зробите цього час від часу, ваш локальний список доступних пакунків може застаріти. Виконуйте цю команду час від часу перед тим як запускати команду dist-upgrade або намагатися знайти новий пакунок. Списки пакунків великі і під час оновлення списків з інтернету будуть завантажені дані розміром в декілька мегабайт.

#apt-get install пакунок — встановлює вказаний пакунок За допомогою цієї команди ви можете отримати останню версію конкретного пакунка і встановити його разом з іншими пакунками, від яких залежить його робота. Якщо запрошуваний пакунок вже встановлений, він буде оновлений до останньої доступної версії.

#apt-get upgrade — оновлює всі програми в системі З часом більшість пакунків програмного забезпечення, які є на вашому комп'ютері, застарівають, так як з'являються нові версії, в яких додаються нові функції та виправляються помилки. Ви можете кожен раз вручну виконувати команду apt-get install foo, але це не дуже зручно, тому apt надає простий спосіб зразу оновити всю систему. Для цього наберіть команду apt-get upgrade і apt перевірить наявність нової версії для кожного пакета у вашій системі, а потім її завантажить та встановить. ця команда ніколи не буде встановлювати нові пакунки, вона призначена тільки для оновлення пакунків, які вже встановлені:

#apt-get remove пакунок — видаляє вказаний пакунок Якщо ви раніше встановили програму і вирішили, що вона вам більше не потрібна, ви можете видалити її за допомогою цієї команди. Оскільки деякі пакунки програм можуть залежати від інших, видалення однієї програми може привести до припинення роботи інших програм. Тому коли ви запускаєте програму apt-get remove, спочатку буде перевірено, чи є інші програми, робота яких залежить від програми, що видаляється і вам буде запропоновано їх також видалити. Це тільки один приклад того, наскільки якісно були розроблені інструментальні засоби управління пакетами, наявні в Ubuntu і завдяки яким ваш комп'ютер не припинить працювати через непрацююче або не повністю встановлене програмне забезпечення. Звичайно, є ймовірність порушити роботу Ubuntu, але, в цілому, ви повинні видалити програми, які пропоновані до видалення. Команда remove також має параметр -- purge, при використанні якого видаляється не тільки сама програма, а й пов'язані з нею конфігураційні файли. #apt-get clean — коли ви просите утиліту apt встановити пакунок програмного забезпечення, вона завантажує пакунок і перш, ніж виконати інсталяцію, зберігає його в кеші на жорсткому диску. Якщо ви потім видаліть пакунок, але пізніше передумаєте і захочете знову його встановити його, утиліті apt не потрібно буде його шукати в інтернеті, оскільки пакунок знаходиться в локальному кеші. Це добре для економії обсягу завантажуваних даних, але через деякий час кеш може займати значний простір на жорсткому диску, так що добре періодично видаляти з нього старі пакунки. Запуск команди apt-get clean повністю очистить кеш пакунків, вивільнивши дорогоцінний дисковий простір. Ця команда абсолютно безпечна, оскільки найбільш погане, що може трапитися, це те, що утиліті apt, можливо, прийдеться завантажити пакунок знову, якщо ви його видалили, а потім знову встановлюєте.

**Apt-cache**

apt-cache — маніпулює кешем доступних пакунків, зазвичай використовується для пошуку пакунка та отримання інформації про нього $apt-cache search keyword — шукає в описах доступних пакунків ключове слово. Використовується для пошуку необхідного програмного забезпечення. $apt-cache show pkgname — виводить повну інформацію про пакунок pkgname, який нас зацікавив.

**Робота з процесами**

Процес — це сукупність програмного кода та даних, які завантажені в пам'ять ЕОМ. На перший погляд процес — це запущена програма (додаток) або команда. Але це не зовсім так. Деякі додатки можуть створювати декілька процесів одночасно. Код процесу не обов'язково має виконуватися в поточний момент часу, так як процес може знаходитися в сплячому стані. В цьому випадку виконання коду такого процесу призупинено. Існує лише 3 стани, в яких може знаходитися процес: Працюючий процес — в даний момент код процесу виконується. Сплячий процес — в даний момент код процесу не виконується, чекає на яку небудь подію (натискання клавіші на клавіатурі, надходження даних з мережі и т.д.) Процес-зомбі — сам процес вже не існує, його код та дані вивантажені з оперативної пам'яті, але запис в таблиці процесів залишається через ті чи інші причини. Кожному процесу в системі призначаються числові ідентифікатори (особисті номери) в діапазоні від 1 до 65535 (PID – Process Identifier – ідентифікатор процесу) та ідентифікатори батьківського процесу (PPID – Parent Process Identifier — ідентифікатор батьківського процесу). PID є ім'ям процесу, за яким ми можемо адресувати процес в операційній системі при використанні різних засобів перегляду і керування процесами. PPID визначає родинні відносини між процесами, які в значній мірі визначають його властивості та можливості. Інші параметри, які необхідні для роботи програми, називають "оточення процесу". Одним із таких параметрів є керуючий термінал — ім'я термінального пристрою, на яке процес виводить інформацію і з якого інформацію отримує. Керуючий термінал мають далеко не всі процеси. Процеси, що не прив'язані до якогось конкретного терміналу називаються "демонами" (daemons). Такі процеси, будучи запущеними користувачем, не завершують свою роботу після закінчення сеансу, а продовжують працювати, так як вони не пов'язані ніяк з поточним сеансом і не можуть бути автоматично завершені. Як правило, за допомогою демонів реалізуються серверні служби, так наприклад сервер друку реалізований процесом-демоном cupsd, а сервер журналювання — syslogd. Для перегляду списку процесів в Linux існує команда ps. Формат команди наступний: ps [PID] [options] — перегляд списку процесів. Без параметрів ps показує всі процеси, які були запущені протягом поточної сесії, за винятком демонів. Options може приймати одне з наступних значень або їх комбінації: -а или -e — показати всі процеси -f — повний лістинг -w — показати повні строки опису процесів. Процеси в ОС Linux володіють тими ж правами, якими володіє користувач, від імені якого був запущений процес. Насправді операційна система сприймає користувача, що в ній працює, як набір запущених від його імені процесів. Адже і сам сеанс користувача відкривається в командній оболонці (або оболонці Х) від імені користувача. Тому коли ми кажемо "права доступу користувача до файлу" то маємо на увазі "права доступу процесів, запущених від імені користувача до файлу".

Батьком всіх процесів в системі є процес init. Його PID завжди 1, PPID — 0. Всю таблицю процесів можна уявити собі у вигляді дерева, в якому коренем буде процес init. Цей процес хоч і не є частиною ядра, але виконує в системі дуже важливу роль — визначає поточний рівень ініціалізації системи і стежить щоб були запущені програми, що дозволяють користувачеві спілкуватися з комп'ютером (mingetty, X або інші). Процеси, імена яких укладені в квадратні дужки, наприклад "[keventd]" — це процеси ядра. Ці процеси керують роботою системи, а точніше такими її частинами, як менеджер пам'яті, планувальник часу процесора, менеджери зовнішніх пристроїв і так далі. Решта процеси є користувацькими, запущеними або з командного рядка, або під час ініціалізації системи. Життя кожного процесу представлена наступними фазами: Створення процесу — на цьому етапі створюється повна копія того процесу, який створює новий. Наприклад, ви запустили з інтерпретатора на виконання команду ls. Командний інтерпретатор створює свою повну копію. Завантаження коду процесу та підготовка до запуску — копія, створена на першому етапі замінюється кодом завдання, яке необхідно виконати і створюється її оточення — встановлюються необхідні змінні і т.п. Виконання процесу Стан зомбі — на цьому етапі виконання процесу закінчилося, його код вивантажується з пам'яті, оточення знищується, але запис у таблиці процесів ще залишається. Вмирання процесу — після всіх завершальних стадій видаляється запис з таблиці процесів — процес завершив свою роботу. Для керування пріоритетом виконання процесу служить команда nice nice — UNIX-утиліта, що запускає програму зі зміненим пріоритетом для планувальника завдань. Якщо не вказано жодного аргументу, команда nice виводить поточний успадкований пріоритет для планувальника завдань. В іншому випадку nice запускає вказану команду зі зміненим пріоритетом. Якщо зсув не вказано, то пріоритет команди збільшується на 10. Привілейований користувач (root) може вказати негативний зсув. Команда nice може зміщати пріоритет в діапазоні від -20 (найвищий пріоритет) до 19 (нижчий пріоритет) від поточного.

Командою, що найбільш часто використовується для керування процесами можна по праву вважати команду kill: kill -SIGNAL pid — надсилає сигнал процесу з ідентифікатором pid. Якщо сигнал не вказаний, команда надсилає процесу сигнал SIGTERM. Ось приклад її використання: kill -SIGKILL 1352 Не менш популярною командою ніж kill є killall: killall -s SIGNAL процес — надсилає сигнал всім процесам з іменем процес. Якщо сигнал не вказаний — надсилає процесу сигнал SIGTERM killall -9 konsole

## Що таке ACL

Список правил ACL ( Access Control List) — це один з основних способів керування безпекою в комп’ютерних мережах та системах.

Він дозволяє адміністраторам **контролювати доступ до конфіденційної інформації** та захищати системи від несанкціонованого доступу та зловживань тими правами доступу, які дозволені.

Також ACL використовуються в різних операційних системах, як-от Windows, Linux та MacOS, а підтримка цього списку правил часто **заздалегідь інтегрована в мережевому обладнанні**, такому як маршрутизатори та комутатори.

## Області застосування списку правил

ACL (Access Control List) можуть застосовуватись у різних сферах, включаючи:

* **Корпоративні мережі.**Адміністратори мереж звертаються до списку правил для керування доступом користувачів та груп до ресурсів мережі. Вони можуть обмежувати доступи до файлів, каталогів, регулювати можливість використання принтерів, електронної пошти, відвідувати вебсайти тощо.
* **Системи керування базами даних.** ACL використовуються для керування доступом до різних частин баз даних, як-от таблиці, уявлення, процедури та тригери.
* **Мережеве обладнання.** ACL застосовуються в маршрутизаторах, комутаторах та іншому мережному обладнанні для керування доступом до мережевих ресурсів, таких як порти, протоколи та сервіси.
* **Хмарні послуги.**ACL задіяні у хмарних сервісах для керування доступом до різних частин та функцій хмарних ресурсів, як-от, наприклад, сховища даних, віртуальні машини, програми та ін.
* **Мобільні пристрої.** ACL можуть застосовуватись у мобільних пристроях, як-от смартфони та планшети, для керування доступом до програм, файлів та інших ресурсів на пристрої.
* **Операційні системи.**Список ACL правил присутній і в ОС (Windows, Linux і MacOS). Він дозволяє контролювати доступ до файлів, папок та інших ресурсів на комп’ютері.

## Особливості ACL

### Гранулярність

Однією з головних особливостей ACL є гранулярність. Вона визначає рівень деталізації прав доступу до ресурсів, який може бути встановлений для користувачів та груп. **Чим вища гранулярність, тим більш детальні права доступу можуть бути встановлені.**

Розрізняють кілька рівнів гранулярності ACL:

* **Рівень об’єкта**. Права доступу призначаються безпосередньо на об’єкти, такі як файли, папки або таблиці баз даних. Наприклад, адміністратор може встановити різні рівні доступу до файлу залежно від користувача чи групи.
* **Рівень атрибуту.**На цьому рівні гранулярність права доступу призначаються на основі атрибутів об’єкта, таких як тип, розмір або розташування. Наприклад, адміністратор може встановити різні рівні доступу до файлів, які містять конфіденційну інформацію порівняно зі звичайними файлами.
* **Рівень операцій.**У цьому випадку права доступу призначаються на основі операцій, які можуть бути виконані над об’єктом, наприклад читання, запис або видалення. Наприклад, адміністратор може встановити різні рівні доступу для користувачів або груп, залежно від того, які операції вони можуть виконувати з файлами.
* **Рівень часу.**Права доступу призначаються на основі часу доступу до об’єктів. Наприклад, адміністратор може встановити різні рівні доступу до файлів залежно від часу дня чи дня тижня.

Хоча більш детальні права доступу встановлені на кожному рівні гранулярності, тим більш точне управління доступом може бути здійснено, більш висока гранулярність може також призвести до більш складного настроювання прав доступу та підвищеного ризику помилок.

### Ієрархія

Друга особливість ACL — ієрархія, коли права доступу можна вказувати на різних рівнях. Наприклад, у файлових системах можна встановити права доступу на рівні файлу, папки, диска або мережного ресурсу.

### Переносність

ACL властива переносимість — правила можна застосовувати до різних типів ресурсів у різних операційних системах та середовищах, у тому числі до файлів, папок, баз даних, мережевих ресурсів тощо.

### Аудит

ACL дозволяє вести аудит — тобто записувати та аналізувати журнали доступу, щоб відстежувати, які користувачі отримують доступ до ресурсів та які операції вони виконують.

### Гнучкість

У списку правил можна визначати права доступу на основі часу доступу, протоколу доступу (наприклад, SSH, HTTP або FTP) та інших факторів, що дозволяє точно налаштувати доступ до ресурсів відповідно до вимог безпеки.

## Види Access Control List

Існує кілька видів Access Control List (ACL), які можуть бути використані для управління доступом до різних ресурсів. Розглянемо деякі найпоширеніші з них:

* **Дискреційний контроль доступу (DAC)**— це різновид ACL, за якого права доступу до ресурсу визначаються власником ресурсу. Наприклад, власник файлу може визначити, які користувачі можуть переглядати, редагувати чи видаляти цей файл.
* **Обов’язковий контроль доступу (MAC)**— це тип ACL, у якому права доступу до ресурсу визначаються системою безпеки з урахуванням політик безпеки. Цей тип ACL зазвичай використовується у високорівневих державних та корпоративних системах, де безпека є пріоритетом.
* **Рольовий контроль доступу (RBAC)**— тип, коли права доступу до ресурсу визначаються з урахуванням ролей, які виконують користувачі. Наприклад, у великій організації адміністратор може створити ролі, такі як «менеджери», «користувачі» та «гості», та визначити права доступу для кожної з цих ролей.
* **Контроль доступу на основі атрибутів (ABAC)**— тип ACL, у якому права доступу до ресурсу визначаються з урахуванням атрибутів користувача чи ресурсу. Наприклад, у системі керування обліковими записами права доступу можуть бути визначені на основі атрибутів, таких як група користувача, дата та час доступу тощо.
* **Контроль доступу на основі політик (PBAC)**— це тип, за якого права доступу до ресурсів визначаються на основі політик безпеки. Ці політики визначають, хто та як може отримати доступ до ресурсу. Вони можуть включати різні параметри, такі як група користувачів, джерело запиту, протокол тощо.

Деякі типи ACL варто розглянути ще ближче.

### Динамічний контроль доступу

Dynamic Access Control (DAC) — це метод керування доступом до ресурсів, який дозволяє динамічно змінювати права доступу на основі умов, що змінюються.

В рамках динамічного контролю доступу створюються динамічні ACL, які визначають права доступу користувачів до ресурсів на основі додаткових умов, таких як час доступу, розташування користувача або його мережевого статусу.

**Приклад динамічного контролю доступу:** система, яка автоматично блокує доступ користувача до конфіденційної інформації, якщо він підключається до мережі з небезпечного місця або використовує пристрій із низьким рівнем безпеки.

Також динамічний контроль доступу може використовуватися для управління доступом до віртуальних ресурсів, таких як віртуальні машини, на основі умов використання, що змінюються.

### Рефлексивний (Reflexive ACL)

Цей метод керування доступом до мережевих ресурсів заснований на аналізі та відстеження мережного трафіку.

Рефлексивні ACL (RACL) дозволяють обмежувати доступ до мережевих ресурсів на основі інформації, що міститься у самому трафіку.

Як правило, RACL використовується для фільтрації трафіку на мережевих маршрутизаторах та брандмауерах. Рефлексивні ACL обробляють пакети, що проходять через пристрої, та аналізують інформацію, яка міститься в заголовках пакетів, щоб визначити, яким чином вони повинні бути оброблені.

RACL можуть фільтрувати трафік, ґрунтуючись на різних параметрах, таких як IP-адреса джерела та призначення, номери портів, протоколи тощо. **Прикладом використання RACL** може бути блокування трафіку від зовнішніх пристроїв, які намагаються встановити з’єднання з внутрішніми пристроями, коли зовнішній трафік не супроводжується внутрішнім трафіком.

Налаштування RACL може залежати від обладнання та мережної архітектури. Зазвичай налаштування RACL включає:

* визначення правил фільтрації на основі параметрів пакетів, як-от IP-адреса джерела та призначення, протоколи та порти;
* встановлення додаткових параметрів, таких як час життя правил та спосіб їх застосування.

### Time-based Access Control List

Time-based ACL (TBACL) — це метод керування доступом, який дозволяє обмежувати доступ до мережних ресурсів на основі часу дня, тижня або місяця.

TBACL використовується для керування доступом до мережних ресурсів у певний час або дня тижня, щоб знизити ризики безпеки та оптимізувати використання мережних ресурсів. Наприклад:

* для керування доступом до мережевих ресурсів під час пікових навантажень;
* для забезпечення безпеки у нічний час або у вихідні дні;
* для запобігання несанкціонованому доступу під час періоду, коли працівники не працюють;
* для дозволу доступу до певних ресурсів лише у певний час.

**Прикладом використання TBACL** може бути обмеження доступу до вебсайту або певних портів у періоди, коли вони не використовуються, наприклад, уночі або у вихідні дні. Це може допомогти знизити ризики безпеки та оптимізувати використання мережевих ресурсів.

TBACL може бути використано разом з іншими методами керування доступом, такими як статичні та динамічні ACL, для створення більш надійних та гнучких мережних систем.

Налаштування TBACL може залежати від використовуваного обладнання та мережевої архітектури. Зазвичай вона включає:

* визначення правил обмеження доступу з урахуванням часу;
* налаштування тимчасових параметрів та встановлення додаткових параметрів, таких як порти, IP-адреси та протоколи.

### ACL у Linux

У Linux Access Control List реалізований як додатковий механізм контролю доступу до файлів і каталогів, який дозволяє гранулярніше управляти правами доступу, ніж стандартна система прав доступу.

Цей інструмент працює на основі розширених атрибутів файлової системи (Extended File System Attributes), які додають додаткові поля для кожного файлу чи каталогу, де зберігаються права доступу для кожного користувача чи групи.

Реалізація ACL у Linux включає дві команди для роботи з ACL:

* setfacl — встановлює ACL для вказаного файлу чи каталогу;
* getfacl — отримує поточні установки ACL для вказаного файлу або каталогу.

Команда setfacl **дозволяє задати права доступу на читання, запис та виконання для певних користувачів чи груп користувачів.**Наприклад, щоб надати користувачеві testuser право на читання та запис файла myfile.txt, потрібно виконати таку команду:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

setfacl -m u:testuser:rw myfile.txt

setfacl -m u:testuser:rw myfile.txt

setfacl -m u:testuser:rw myfile.txt

Команда getfacl дозволяє **отримати поточні установки ACL для вказаного файлу або каталогу.** Наприклад, щоб дізнатися, які ACL встановлені для файла myfile.txt, потрібно виконати таку команду:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

getfacl myfile.txt

getfacl myfile.txt

getfacl myfile.txt

## Приклад налаштування статичного Access Control List (ACL)

Розглянемо приклад налаштування статичного Access Control List (ACL) на маршрутизаторі Cisco для обмеження доступу до вебсервера (IP-адрес 192.168.1.10) мережі 192.168.0.0/24 тільки для окремих IP-адресов.

Визначаємо номер ACL та створюємо його:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

Router(config)# access-list 10 permit any

Router(config)# access-list 10 permit any

Router(config)# access-list 10 permit any

Додаємо правило для дозволу доступу до вебсервера лише для обраних IP-адрес. У цих командах ми дозволяємо доступ до вебсервера лише з IP-адрес 192.168.2.10 і 192.168.2.11:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

Router(config)# access-list 10 permit tcp host 192.168.2.10 host 192.168.1.10 eq 80

Router(config)# access-list 10 permit tcp host 192.168.2.11 host 192.168.1.10 eq 80

Router(config)# access-list 10 permit tcp host 192.168.2.10 host 192.168.1.10 eq 80 Router(config)# access-list 10 permit tcp host 192.168.2.11 host 192.168.1.10 eq 80

Router(config)# access-list 10 permit tcp host 192.168.2.10 host 192.168.1.10 eq 80

Router(config)# access-list 10 permit tcp host 192.168.2.11 host 192.168.1.10 eq 80

Усі інші запити до вебсервера (з усіх IP-адрес, крім тих, які ми дозволили в попередніх правилах) відхиляємо:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

Router(config)# access-list 10 deny tcp any host 192.168.1.10 eq 80

Router(config)# access-list 10 deny tcp any host 192.168.1.10 eq 80

Router(config)# access-list 10 deny tcp any host 192.168.1.10 eq 80

Застосовуємо ACL до інтерфейсу FastEthernet 0/0 вхідним напрямком, щоб обмежити доступ до вебсервера лише для тих IP-адрес, які ми дозволили в правилах ACL.

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

Router(config)# interface FastEthernet 0/0

Router(config-if)# ip access-group 10 in

Router(config)# interface FastEthernet 0/0 Router(config-if)# ip access-group 10 in

Router(config)# interface FastEthernet 0/0

Router(config-if)# ip access-group 10 in

Це приклад налаштування статичного ACL на маршрутизаторі Cisco. Він може змінюватись в залежності від використовуваного обладнання та мережевої архітектури. Важливо пам’ятати, що **неправильне налаштування ACL може призвести до блокування трафіку або порушення безпеки мережі**, тому рекомендується ретельно перевіряти правила ACL перед застосуванням.

## Приклад скрипта автоматизації налаштування Access Control List (ACL)

Цей приклад скрипта на Python дозволяє створювати статичний ACL для дозволу або блокування трафіку з певних IP-адрес та/або з певних портів. Скрипт призначений для використання на маршрутизаторах Cisco із встановленою бібліотекою Netmiko, яка дозволяє керувати обладнанням через SSH-протокол:

Plain text

Copy to clipboard

Open code in new window

EnlighterJS 3 Syntax Highlighter

from netmiko import ConnectHandler

# дані для підключення до маршрутизатора

device = {

'device\_type': 'cisco\_ios',

'ip': '192.168.1.1',

'username': 'admin',

'password': 'password',

'port': 22,

}

# список IP-адрес та портів для дозволу або блокування

allowed\_ips = ['192.168.2.10', '192.168.2.11']

blocked\_ports = [22, 23]

# створення та застосування ACL

acl\_name = 'ACL1'

acl\_rules = []

acl\_rules.append('access-list ' + acl\_name + ' remark Allow or block specific IP addresses and ports')

acl\_rules.append('access-list ' + acl\_name + ' permit ip ' + ', '.join(allowed\_ips) + ' any')

for port in blocked\_ports:

acl\_rules.append('access-list ' + acl\_name + ' deny tcp any any eq ' + str(port))

acl\_rules.append('access-list ' + acl\_name + ' permit ip any any')

# підключення до маршрутизатора та виконання команд налаштування ACL

with ConnectHandler(\*\*device) as ssh:

ssh.enable()

for rule in acl\_rules:

ssh.send\_command(rule)

ssh.send\_command('interface FastEthernet 0/0')

ssh.send\_command('ip access-group ' + acl\_name + ' in')

from netmiko import ConnectHandler # дані для підключення до маршрутизатора device = { 'device\_type': 'cisco\_ios', 'ip': '192.168.1.1', 'username': 'admin', 'password': 'password', 'port': 22, } # список IP-адрес та портів для дозволу або блокування allowed\_ips = ['192.168.2.10', '192.168.2.11'] blocked\_ports = [22, 23] # створення та застосування ACL acl\_name = 'ACL1' acl\_rules = [] acl\_rules.append('access-list ' + acl\_name + ' remark Allow or block specific IP addresses and ports') acl\_rules.append('access-list ' + acl\_name + ' permit ip ' + ', '.join(allowed\_ips) + ' any') for port in blocked\_ports: acl\_rules.append('access-list ' + acl\_name + ' deny tcp any any eq ' + str(port)) acl\_rules.append('access-list ' + acl\_name + ' permit ip any any') # підключення до маршрутизатора та виконання команд налаштування ACL with ConnectHandler(\*\*device) as ssh: ssh.enable() for rule in acl\_rules: ssh.send\_command(rule) ssh.send\_command('interface FastEthernet 0/0') ssh.send\_command('ip access-group ' + acl\_name + ' in')

from netmiko import ConnectHandler

# дані для підключення до маршрутизатора

device = {

    'device\_type': 'cisco\_ios',

    'ip': '192.168.1.1',

    'username': 'admin',

    'password': 'password',

    'port': 22,

}

# список IP-адрес та портів для дозволу або блокування

allowed\_ips = ['192.168.2.10', '192.168.2.11']

blocked\_ports = [22, 23]

# створення та застосування ACL

acl\_name = 'ACL1'

acl\_rules = []

acl\_rules.append('access-list ' + acl\_name + ' remark Allow or block specific IP addresses and ports')

acl\_rules.append('access-list ' + acl\_name + ' permit ip ' + ', '.join(allowed\_ips) + ' any')

for port in blocked\_ports:

    acl\_rules.append('access-list ' + acl\_name + ' deny tcp any any eq ' + str(port))

acl\_rules.append('access-list ' + acl\_name + ' permit ip any any')

# підключення до маршрутизатора та виконання команд налаштування ACL

with ConnectHandler(\*\*device) as ssh:

    ssh.enable()

    for rule in acl\_rules:

        ssh.send\_command(rule)

    ssh.send\_command('interface FastEthernet 0/0')

    ssh.send\_command('ip access-group ' + acl\_name + ' in')

Ми вказуємо список IP-адрес, яким дозволено доступ, та список портів, які мають бути заблоковані. Потім ми створюємо ACL за допомогою списку правил та застосовуємо його до інтерфейсу вхідним напрямком.

Цей приклад скрипту можна легко доопрацювати та адаптувати під конкретні вимоги, залежно від того, які правила ACL потрібно налаштувати.

**1.   Принципи** **тайм-менеджменту**

Тайм-менеджмент представляє особливу складність для системних адміністраторів, оскільки наші проблеми унікальні (суміш проектів і переривань), наші технічні наставники не особливо сильні в тайм-менеджменті, а наші далекі від техніки начальники нічого не тямлять в нашій роботі. Однак у нас в активі то, що ми самі технарі і для організації свого часу можемо застосовувати технологічні рішення.

• Зовнішні переривання (клієнти) і переривання, які ми самі собі нав'язали (інтернет-пейджер, повідомлення про нові електронні листи і т. Д.), Згубні для продуктивності. Повернення від переривання до роботи вимагає часу і загрожує помилками.

• Спільно з колегами організуйте взаємну захист від переривань, щоб хтось із них обробляв переривання в той час, коли вам необхідно працювати над проектом.

• Всю інформацію по тайм-менеджменту зберігайте в єдиній базі даних. Коли все знаходиться в одному місці, організованість дається легше.

• Бережіть свій мозок для важливої інформації. Використовуйте його безпосередньо для роботи, а списки справ, зустрічей і просто замітки записуйте в органайзер.

• Розробляйте процедури і дотримуйтеся їх. Замість того щоб періодично винаходити велосипед або повторювати процес прийняття рішень, організуйте робочі процедури.

• Виробляйте звички і заклинання. Вони дозволять вам повторно застосовувати вдалі рішення.

• Зберігайте зосередженість під час роботи над проектом. Ви будете працювати краще, якщо зосередитеся на чомусь одному.

• У повсякденному житті користуйтеся тими ж прийомами, що і на роботі. Ваше життя поза офісом безумовно важлива, і її слід організувати тими ж методами, що і професійну діяльність, а то ви забудете про відпочинок і розваги.

**Фішки**

Кожен готовий дати пораду, як уникнути відкладання справ в довгий ящик. В Google ви знайдете десятки тисяч посилань. Ви знайдете таблиці, складені в університетських консультаційних центрах, які допомагають вам розібратися з факторами, що заважають вашій роботі. Ви знайдете книги і статті. Ви знайдете списки з десятками причин, які змушують людину день у день відкладати якусь справу, і дізнаєтеся, як з цим боротися. ви знайдете професійних «вчителів життя», які (особисто або по телефону) навчать вас долати цю та інші життєві проблеми (за плату). Вибирайте, що вам більше підходить. Зробіть це прямо зараз.

Я думаю, ви повинні перш за все запам'ятати, що відкладати справа «на потім» природно. Така поведінка відбувається з боязні і невпевненості в собі. Ми всі боїмося змін. Ми всі сумніваємося в своїй здатності досягти результату.

Замість того щоб концентруватися на невпевненості в собі, перенесіть увагу на протилежний об'єкт - ваше самоповагу. Самоповага подібно стосі покерних фішок. Якщо у вас мало фішок, ви можете робити лише маленькі ставки.

Це означає, що ви не зможете виграти купу грошей. Вам доведеться часто пасувати, щоб не втратити останню фішку. Коли у вас багато фішок, ви можете робити великі ставки, здатні принести великий виграш. Ви можете ризикувати. ви можете робити те, на що ніколи б не зважилися з невеликим кількістю фішок. Ви можете виграти!

При низької самооцінки ми з більшою ймовірністю кинемо розпочату справу або навіть не приступимо до нього. Не намагаючись, нічого не доб'єшся. Ось ми і не добиваємося. При високій самооцінці ми легше йдемо на ризик і ставимо себе в становище, в якому можемо виграти.

Тепер зрозуміло, чому я порівняв самоповагу з великою стопкою фішок?

Магія в тому, що фішки самоповаги існують тільки у вашій свідомості, і ви можете створити їх скільки завгодно!

У покері фішки є реальними фізичними об'єктами. Ви не можете клацнути пальцями так, що з'являться нові фішки. Зате ви можете виконати якийсь ритуал, щоб чарівнимчином збільшилася кількість фішок вашого самоповаги. Моє заклинання «Краще раніше, ніж пізніше» дає мені фішки, що допомагають не відкладати справи на потім. прояви любові улюбленим вами людиною дає вам нові фішки. мовчазна підтримка колег, що дозволяє вам читати цю книгу, породжує нові фішки. Допомагає все, від чого зростає кількістьваших фішок. Якщо гучні вигуки «Я можу!» Додають вам фішок самоповаги, кричіть, скільки хочете.

Дуже просто, так?

Фахівці сходяться на думці: покупка цієї книги, так би мовити, «автоматично» збільшує кількість наявних у вас фішок. Переверніть сторінку і приступайте до роботи.

**2.   Зосередженість як засіб** **проти переривань**

**Вам важко заснути?**

Заснути - все одно що дати мозку розслабитися. Як ви можете розслабитися, якщо вам треба пам'ятати про завтрашні справи? У вас не вийде і те й інше одночасно.

Покладіть поруч з ліжком ручку і блокнот. Якщо щось не дає вам заснути, зробіть запис у блокноті і знову попітайтесь заснути. Готовий посперечатися, ви скоро заснете.

Цей прийом працює і тоді, коли ми чимось стурбовані або розсерджені. Занепокоєння не дає нам заснути, тому що ми намагаємося не забути зробити щось, що відноситься до предмету занепокоєння. Гнів не дає нам заснути, тому що ми намагаємося не забути про те, що розсердилися. Якщо ви запишете свої неспокійні або гнівні думки, ваш мозок зможе розслабитися, тому що ви знаєте: вранці блокнот буде поруч.

Ще краще, якщо ви зателефонуєте собі на роботу і залишите повідомлення на автовідповідачі. Цей прийом можна застосувати скрізь, де є телефон. І він виключає для вас шанс забути блокнот будинку!

У багатьох стільникових телефонів і МРЗ-плеєрів є вбудований диктофон. Заведіть звичку користуватися диктофоном, щоб не забути вдалі ідеї.

• Важливо підтримувати зосередженість. Зосередженість досягається за допомогою усунення відволікаючих чинників і ефективної обробки переривань.

• Переривання, по суті, є спробою інших людей керувати вашим часом. Переривання - споконвічний ворог зосередженості і, отже, тайм-менеджменту.

• Переривання завдають шкоди, оскільки вони затримують виконання поточної роботи, а повернення до неї значною кількістю помилок з вашого боку. Виправлення цих помилок може зайняти більше часу, ніж сама робота.

• Усунення відволікаючих чинників дозволяє вам підтримувати зосередженість. Приберіть все зайве з робочого столу і з екрана комп'ютера. Приберіть відволікаючі вас предмети з робочою кому нати. Вимкніть інтернет-пейджери, повідомлення про нові електронні листи і т. П.

• У кожного свій період піку розумової і фізичної активності. З'ясуйте, в який час настає ваш пік, і відповідно плануйте робочий день.

• Перша година робочого дня може виявитися найпродуктивнішим, оскільки у вас буде мінімум переривань. Прийшовши на роботу трохи раніше колег, ви зможете збільшити цей відрізок. Не витрачайте це час на обслуговування системи; скористайтеся ним для роботи над проектами.

Правило «Делегуйте, реєструйте або дійте» дозволяє вам керувати своїм часом. Дотримуйтеся його, коли вашу роботу переривають. Делегування завдання означає доручення її іншого співробітника. Реєстрація дозволяє вам зафіксувати питання, але вирішити його в більш зручний час. дія неминуче лише в самому крайньому випадку, наприклад при відмові системи.

• Реєструючи запит, ви отримуєте можливість планувати своє час, не залежачи від переривань. Ми обговоримо цю тему в розділі 8.

• Беручи запит клієнта, ви повинні всім своїм виглядом демонструвати це. Переконайтеся, що клієнт бачить, як ви реєструєте запит, і отримаєте його підтвердження.

• Клієнт віддає перевагу знати напевно, що його запит прийнятий (навіть якщо він буде виконаний пізніше), а не гадати, чи дійшов він до вас.

• Система реєстрації клієнтських запитів, така як RT, дозволяє вам зберігати їх в єдиній базі даних, щоб інші системні адміністратори могли їх обробляти, а клієнти - перевіряти поточний статус своїх запитів.

• Ніколи не покладайтеся на пам'ять, якщо клієнт звертається до вас із запитом. Записуйте запит на папері або в електронному вигляді. Мозок знадобиться вам для більш важливої роботи.

**Деякі загальні поради**

На жаль, в цій книзі неможливо дати конкретні рекомендації про те, як ви повинні працювати. Я навіть не знаю, яка операційна система у вас встановлена. Проте я можу порадити наступне:

• Сім разів відміряйте, один раз відріжте. Перш ніж зробити незворотні дії, оціните всі наслідки.

• Створення резервної копії до того, як змініть файл. Резервна копія позбавить вас від багатьох проблем. Однак створювати її слід до внесення змін!

• Якщо нічого не вдіє, почитайте документацію. якщо вам не вдається знайти рішення, зверніться до джерел, про які ви часто забуваєте.

• Під час налагодження вносите тільки одна зміна за раз. Вносячи тільки одна зміна, ви бачите, що вплинуло на систему. Так вам вдасться уникнути плутанини в процесі налагодження.

• Завжди тестируйте свою роботу. Деякі люди, здається, ніколи не помиляються. Однак я виявив, що вони ретельно тестують те, що зробили; просто ми цього не бачимо.

• Робота не закінчена, поки її не протестує клієнт. вам може здаватися, що ви зробили все, як треба, але поки клієнт не перевірив вашу роботу, ви не зможете дізнатися, чи вирішили ви його проблему.

• Найдивніші проблеми часто походять від невірної конфігурації DNS. DNS впливає на роботу багатьох підсистем, часто самим незрозумілим чином. нерідко проблема з DNS маскується під іншу проблему. Це справедливо як щодо клієнтських комп'ютерів, яким недоступні їх DNS-сервери, так і по відношенню до клієнтів, які намагаються досягти адресата, користуючись помилковою інформацією, отриманою від DNS.

**Головне почати**

Почавши якусь справу, ви зрозумієте, що все не так складно, як очікувалося. Ми часто не беремося за роботу, шукаючи виправдання в тому, що у нас немає на неї часу. Однак, приступивши до дій, бачимо, що вона виконується досить швидко.

Одна знайома обіцяла мені читати рукопис цієї книги і ділитися враженнями, але її відгук про першому розділі прийшов на кілька тижнів пізніше, ніж я очікував. Вона відкладала цю справу, тому що переконала себе, що їй потрібно безперервний двогодинний відрізок часу, щоб виконати роботу як слід. Виявилося, що в першому розділі менше десяти сторінок, і знайома прочитала її всю за півгодини.

Якби вона просто почала читати, а не міркувала, коли краще почати, то впоралася б набагато швидше.

**3.   Рутинні процедури**

Хороша процедура дозволяє вам економити зусилля і скорочує час на прийняття рішень.

• Процедури дають вам можливість «один раз подумати і багато разів зробити ».

• Розробіть процедуру, згідно з якою ви будете завжди записувати призначені зустрічі і необхідні справи в органайзер і всюди носити його з собою.

Чим більше процедур ви разработаете, тим менше розумової енергії ви будете витрачати на рішення незначних питань і тим краще ви зможете зосередитися на творчій стороні вашої професії. Перегляньте свій робочий день на предмет можливості створення власних процедур. Покажчиками на такі можливості є:

• Події, що повторюються, не включені в розклад.

• Завдання технічного обслуговування.

• Підтримка взаємин і робочої обстановки.

• Ситуації, в яких роздуми про якійсь справі забирають більше часу, ніж воно саме.

Речі, які ви часто забуваєте.

• Одноразові або фонові завдання, які не можна все час ігнорувати.

• Підвищення професійного рівня.

• Інформованість про новинки.

Заплануйте час на читання

Ви не встигаєте прочитати все комп'ютерні журнали, на які підписалися? Заплануйте один годину на читання щотижня. Знайдіть затишне місце і читайте, скільки зможете.Викидайте те, що не вдалося прочитати; цікавтеся тільки найсвіжішою інформацією.

**4.   Система Цикл**

Необхідно вміти реєструвати всі клієнтські запити і доводити їх виконання до кінця (або відхиляти). Клієнти (ті, кого ви обслуговуєте) і менеджери (ті, хто вирішує, коли підвищувати вам зарплату) цінують це вміння, тому що хочуть, щоб їхні запити і проекти виконувалися, а не «спускалися на гальмах».

• Хороший «завершальний кидок» - ключ до підвищення по службі.

• Не дайте запитом померти в зародку - реєструйте їх все без винятку. Якщо клієнт зустріне вас в коридорі і попросить про щось, не покладайтеся на свою пам'ять. Якщо ви не можете відразу записати прохання, запропонуйте клієнту відправити її вам по електронній пошті або в систему реєстрації клієнтських запитів. Так ви перекладете відповідальність за реєстрацію прохання на клієнта, що не ризикуючи забути про неї.

• Ніщо так не ображає, що не злить і не розчаровує клієнта, як обіцянку, дану системним адміністратором і потім забуте.

• Чим більше справ ви повинні зробити, тим важче вам організувати свою роботу. Починаючи з якогось моменту ви будете витрачати більше часу на організацію роботи, ніж на її виконання.

• Щоб не забувати запити, реєструйте їх надійним способом. Людський мозок не так надійний, як папір або електронні пристрою. Записуйте запити в той момент, коли отримуєте їх. Завжди записуйте всі запити. Бережіть свій мозок для болем важливих завдань.

Системи зі списками справ не працюють по ряду причин. Безладно розкидані записи губляться. Єдиний список справ перетворюється в гнітючий Нескінченний Фатальний Список. Обидві системи негативно впливають на самооцінку.

• У системі Цикл використовуються календар зустрічей і важливих дат, список довгострокових цілей, список справ на сьогодні (і на кожен день) і розклад на сьогодні, що дозволяє планувати роботу.

• Кожен день починається з десятихвилинної планування. Вивчіть календар, щоб зрозуміти, скільки часу у вас піде на зустрічі і наради. Решту часу ви можете присвятити справах з сьогоднішнього списку. Ви визначаєте, чи достатньо у вас часу для виконання всіх справ зі списку, і, якщо буде потрібно, переносите низькопріоритетні справи на інші дні або ведете переговори з клієнтами про перенесення термінів.

**Почніть прямо зараз!**

Коли ви почнете дотримуватися системи Цикл, вона здасться вам незручною. Однак з часом ви відчуєте себе більш комфортно. Ви пристосуєте її під свій стиль, коли краще в ній розберетеся. Психологи стверджують, що для формування звички потрібно 21 день. Протягом 21 дня наполегливо робіть одне і те ж, щоб ваш мозок почав ставитися до цієї справи як до звички, що не вимагає особливих зусиль. моє правило "список справ на кожен день" не стане вашою звичкою, поки ви не почнете його дотримуватися. Заплануйте до виконання вашу нову звичку на сьогодні. Решта 20 днів вам буде набагато простіше.

**5.   Система Цикл:** **списки справ і розкладу**

• Протягом дня виконуйте намічений план. Відзначайте виконані пункти хрестиком, а перенесені - рискою.

• Ближче до кінця робочого дня обробіть всі невиконані пункти, щоб затримки у виконанні запитів були неприємним сюрпризом для клієнтів.

• В кінці дня всі пункти виявляються обробленими в тому сенсі, що вони або завершені, або якимось чином перенесені на наступні дні. Мета цієї процедури в тому, щоб залишити офіс ні з відчуттям непосильної ноші, а з усвідомленням того, що всі завдання під контролем. Ви можете піти додому з посмішкою, знаючи, що впоралися з сьогоднішніми справами. Завтра буде нова робота.

• Якщо ви переробили все справи до кінця робочого дня, нагородите себе. Ви можете зайнятися своїм особистим проектом, почитати технічну літературу, навести порядок в приміщенні, створити заділ на Наступного дня і т. д. Якщо у вас гнучкий графік, можна піти додому раніше.

• Якщо протягом дня з'являються нові справи, їх слід запланувати на завтра, або доведеться перетрусити сьогоднішнє розклад, щоб виконати їх сьогодні. Якщо по ходу робочого дня постійно виникають справи, які треба зробити «прямо зараз», можна вже при плануванні відвести певний відрізок часу на обробку цих переривань.

• Управляти особистими справами можна по цій же системі. застосовуючи єдину систему для роботи та особистого життя, ви, по-перше, практикуєтеся в системі Цикл, по-друге, носите з собою тільки один інструмент тайм-менеджменту, нарешті, не забуваєте про особисті справах, від чого ваше життя робиться світліше.

• Важливо не те, якої системи ви дотримуєтеся, а то, що це система. Іншими словами, система Цикл хороша для мене, але якщо вас влаштовує інша, застосовуйте її. Правда, якщо ви читаєте цю книгу, ймовірно, у вас немає власної системи. Почніть застосовувати систему Цикл, а потім адаптуйте її під свої привички і стиль життя.

• Систему Цикл можна реалізувати за допомогою портативного комп'ютера або паперового органайзера. У кожного інструменту є свої плюси і мінуси. Спробуйте обидва і виберіть той, що вам більше підходить.

**6.   Система Цикл: робота з календарем**

• Ведення календаря важливо для вас і вашої кар'єри. Пунктуальність асоціюється з відповідальністю і надійністю. Люди не є на зустрічі і забувають про нарадах, не отримують підвищення по службі.

• Без добре організованого календаря ви ризикуєте пропустити важливі події як по роботі, так і в особистому житті. необхідно дотримуватися балансу між роботою, сім'єю, громадською діяльністю, особистими проектами, сном і т. д. Ваш календар допоможе вам в цьому.

• У записнику повинна бути секція для запису зустрічей (нарад та інших подій). Записуйте дату і час зустрічі, як тільки домовтеся про неї. Не погоджуйтеся на зустріч, поки не звіряючись з календарем.

• Ваш календар є складовою частиною системи Цикл. Ви записуєте в нього зустрічі, наради, важливі події та іншу інформацію. Плануючи свій день, почніть з календаря, щоб скласти розклад і доповнити список справ на сьогодні.

• У паперовому органайзері можна записувати події в графи календаря за певною системою. Я спочатку вписую дні народження і річниці, потім відпустки і багатоденні заходи.

В середині графи календаря я пишу міні-розклад на цей день: ранкові зустрічі, обідня перерва, зустрічі в другій половині дня. Нижню частину графи я відводжу під плани на вечір.

• Домовляючись про зустріч, враховуйте свій ритм активності. якщо є вибір, плануйте розумову роботу на час піку своєї розумової активності.

• Складаючи спільні плани, завжди звіряйтеся з календарем, перш ніж погодьтеся на зустріч. Чи не дивуйтеся тим, що вашому співрозмовнику доводиться почекати, поки ви дістанете і відкриєте органайзер.

• Користуйтеся автоматичними засобами нагадування про зустрічі. Встановіть подачу сигналу на портативному комп'ютері. якщо у вас паперовий органайзер, використовуйте альтернативну технологію (наприклад, будильник).

• В паперовий органайзер можна записувати повторювані події на основі списків щотижневих, щомісячних і щорічних подій. У понеділок запишіть в календар все щотижневі зустрічі. У перший день місяця запишіть щомісячні зустрічі. А в перший день нового року - щорічні події.

• У більшості фірм є щорічний ритм діяльності. Наприклад, найбільш напружений час у торгуючих підприємств припадає на грудень. З'ясувавши цей ритм, ви зможете планувати свої проекти відповідно до нього. В іншому випадку ви будете гребти проти течії. Якщо у вашої фірми немає чітко вираженого ритму, визначте ритм для себе.

**7.   Система Цикл: довгострокові цілі**

• Для досягнення довгострокових цілей вам потрібно сформулювати їх і діяти за відповідними напрямами.

• Чи не записавши свої цілі, ви так і будете тупцювати на місці в надії, що вам пощастить.

• Цілі мають бути вимірюваними: результат повинен бути відчутним або виражатися в цифрах так, щоб інша людина могла його перевірити.

• У цілей повинні бути вказані терміни: якщо ви знаєте, коли потрібно досягти мети, це допомагає вам дотримуватися певного ритму.

• Почніть з перерахування цілей, яких ви хочете досягти на роботі і в особистому житті через місяць, через рік і через п'ять років. Розставте пріоритети. Складіть список кроків, необхідних для досягнення ваших цілей. Занесіть наступний крок для кожної цілі в список справ на майбутнє. Раз на місяць переглядайте мети і кроки, міняйте пріоритети, якщо треба, і заносите наступні кроки в список справ.

• Виконуйте наступні кроки, як будь-які інші справи зі списку. Згодом кожна мета буде досягнута або буде під контролем.

• Регулярно переглядайте список цілей. Додавайте нові й видаляйте досягнуті. Обробляйте кроки відповідним чином .

**8.   Розстановка пріоритетів**

• Якщо у вас багато роботи, розстановка пріоритетів набуває величезне значення. Правильні пріоритети ще більш важливі, якщо у вас фізично не вистачає часу на всі справи. Якщо ви не дуже завантажені, годиться будь-яка схема розстановки пріоритетів.

• Виконання справ в порядку їх слідування в списку - хороший підхід, коли справ небагато. Оскільки більш старі справи Обачним «спливають» до початку списку, ви їх неодмінно виконайте. якщо ви не знаєте, яке діло виконувати наступним, приступайте до наступного по порядку. Краще випо лніть перша справа в списку, ніж втрачати час на роздуми про те, яке з цих справ робити першим.

• Розстановка пріоритетів відповідно до очікувань клієнтів на увазі, що починати слід з тих справ, які, по мені нию користувачів, займають мало часу. Клієнти вважають, що швидко повинні виконуватися ті незначні запити, через які затримуються великі проекти. Розставляючи пріоритети по цій схемі, ви витратите стільки ж робочого часу, скільки і при будь-якій іншій, а ваші клієнти будуть задоволені.

• Призначаючи одному з проектів найвищий пріоритет, засновуйте свій вибір на ефекті. Проект з максимальним ефектом, що вимагає значних зусиль для його завершення, корисніше для вас, ніж більш простий проект з малим ефектом.

• Запитам, що виходить від вашого шефа, слід призначити особливий пріоритет. Вони нерідко викликані причинами, про які ви і не підозрюєте. Не допускайте, щоб через вас затримався його болем важливий проект!

• Щоб керувати своїм шефом, ви повинні робити три речі: доводити до його відома свої кар'єрні цілі; делегувати наверх тільки ті завдання, які вимагають застосування його повноважень; розуміти його цілі і сприяти їх досягненню. Роблячи це, ви краще погодите зі своїм шефом, а він стане більш гнучко реагувати на ваші запити, знаючи, що ви дієте в його інтересах.

**9.   Як впоратися зі стресом**

• Системне Адміністрування- робота, сполучена зі стресом. Отже, вміння впоратися зі стресом має входити в число ваших професійних навичок.

• Нерідко найсильнішими джерелами стресу є відчуття перевантаженості роботою або конфліктуючі вказівки від кількох начальників, по-різному розставляють пріоритети.

• Справитися з відчуттям перевантаженості добре допомагає технологія тайм-менеджменту, особливо система Цикл.

• Проблему стресу, викликаного конфліктуючими пріоритетами, слід припиняти в корені. Для цього необхідно провести переговори з джерелами конфлікту або звести зацікавлені сторони для діалогу.

• Релаксацією можна управляти, і цьому треба вчитися. Вам слід виділити в своєму графіку час для цілеспрямованої релаксації за допомогою йоги, медитації, масажу або аналогічних методів.

• Ви повинні навчитися прийомам релаксації, щоб досягти позитивних результатів. Ви повинні запланувати свідому релаксацію, так само як плануєте заміну масла в машині.

• Відпустка - це не подарунок вам від роботодавця. Це частина вашого режиму релаксації і програми зняття стресу. Ви йдете у відпустку, щоб підтримати своє психічне здоров'я. Відмовитися від чергової відпустки - все одно, що відмовитися від чергового чищення зубів. Вам може здаватися, що на цей раз у відпустку можна не йти, але в кінцевому рахунку це вдарить по вашому здоров'ю. Якщо ви розтратите час відпустки на повсякденні домашні справи, такий відпустку не допоможе вам впоратися зі стресом.

• Я не люблю каркати, але хороший сисадмін допускає, що якщо його завтра зіб'є вантажівка, фірма зможе продовжити працювати і без нього. Великий відпустку - прекрасний спосіб без шкоди для здоров'я протестувати план заміщення системного адміністратора і якість документації. Можливо, вам вдасться знайти і усунути прогалини в плані заміщення, коли ви повернетеся з одно- або двотижневого відпустки.

• Влаштуйте собі перерву. Глибоко вдихніть. Ніхто не говорив, залишаючи цей світ: «Як шкода, що я провів на роботі так мало часу». Розслабтеся, все це тільки нулі і одиниці.

**10.        Управління електронною поштою**

• Більшість системних адміністраторів отримує більше електронних листів, ніж вони можуть обробити. Якщо ви не керуєте своєю електронною поштою, вона починає керувати вами. Візьміть контроль над електронною поштою в свої руки, і це буде величезним кроком на шляху до повного контролю над вашим часом.

• Папка Вхідні - невдале місце для ведення списку справ.

• В ідеалі папка Вхідні повинна бути порожньою. Тому всі ваші дії з черговим надійшли листом повинні завершуватися або видаленням цього листа, або збереженням його в іншій папці. Для досягнення цього ідеалу я пропоную вам метод, що припускає обробку кожного листа за допомогою однієї з наступних операцій: фільтрація, видалення без читання, читання і виконання якоїсь дії, виконання запиту і видалення листи.

• Фільтрація. Користуйтеся програмної фільтрацією для попередньої обробки електронної пошти і автоматизації багатьох завдань.

• Видалення без читання. Деякі типи листів можна спокійно видаляти, не читаючи.

• Читання і виконання дії. Якщо лист потрібно прочитати, переслати, зареєструвати (в записнику або системі реєстрації запитів) або зберегти в певній папці, виконайте потрібні дії і видаліть лист з папки Вхідні. Не дозволяйте йому там залежуватися.

• Виконання запиту і видалення листи. Якщо міститься в листі запит вимагає декількох хвилин, виконайте його тут же і видаліть лист.

• Якщо ви задумали розгребти купу листів, що скупчилися на даний момент, я рекомендую вам зберегти її де-небудь і забути про неї. Якби якийсь з цих листів містила дійсно важнуюінформацію, вона дійшла б до вас іншим шляхом. Електронна пошта є ефемерною. Чим старше лист, тим меншу цінність воно представляє.

**11.       Виключення непродуктивних витрат часу**

• Непродуктивні витрати часу - це діяльність з низьким співвідношенням корисного результату і витраченого часу. Замість того щоб оптимізувати подібну діяльність, спробуйте зовсім відмовитися від неї.

• Деякі справи можуть розтягуватися, заповнивши собою весь ваш час. Ви можете перервати їх, якщо встановите для себе якийсь часовий ліміт. Виробіть обмежувальні правила, наприклад: «Починаючи [таке-то справа], я встановлю таймер, який подасть звуковий сигнал через 10 хвилин».

• Ніщо так не допомагає побороти спокусу, як присутність свідка. Працюючи в одному кабінеті з колегою, ви менше схильні відволікатися від роботи на сторонні справи.

• У сучасному житті є багато факторів, що призводять до непродуктивних витрат часу. Це заплановані необов'язкові справи, підписка на розсилки по електронній пошті, інтернет-чати, сторонні дискусії в робочий час, настирливі торгові агенти і вербувальники, процеси, що виконуються вручну, але підлягають автоматизації, і т. Д. Розпізнавши подібний фактор, позбудьтеся від нього.

• Удома ви теж можете боротися з такими факторами за допомогою цифрового відеомагнітофона для перегляду тільки важливих телепередач, списку фільмів, які хочете взяти напрокат, і служби Netflix, щоб менше часу проводити в магазинах відеопродукції; користуйтеся послугами пралень і фірм побутового обслуговування, щоб звільнити більше часу для насолод.

• Вивчення процесу установки системи, яка встановлюється тільки один раз, малокорисні. Наприклад, в разі централізованої установки системи резервного копіювання виділення з бюджету коштів на послуги фахівця обернеться значним виграшем.

• З точки зору системного адміністратора непродуктивною витратою часу є будь-яка справа, яке можна було б виключити, якби знайшовся час на створення інфраструктури, що дозволяє від цієї справи відмовитися. Іншими словами, найкращим засобом тайм-менеджменту для системного адміністратора є хороша програмно-апаратна інфраструктура. Мислячи стратегічно, ви зможете виключити тактичні завдання в довгостроковій перспективі. Іншими словами, ви зможете перестати підтирати калюжі на підлозі, полагодив кран.

**12.        Документація**

• Сховище документів - прекрасний інструмент тайм-менеджменту.

• Сховище, орієнтоване на клієнтів, надає їм необхідну інформацію, і вони менше турбують вас.

• Місце для внутрішньої ІТ-інформації дозволяє вам організувати довідковий матеріал для економії часу згодом. Особливо корисними можуть виявитися технологічні карти - короткі замітки про те, як слід виконувати складні процедури, - вашим колегам не доведеться винаходити велосипед.

• Добре задокументовану процедуру легше делегувати іншого співробітника. Це дозволить вам викреслити пункт зі свого списку справ, доручивши цю процедуру комусь іншому.

• Технологія Wiki полегшує доступ до сховища і не вимагає знання HTML. Дозволивши всім користувачам редагувати (майже) всі сторінки, ви досягнете того, що сторінки будуть містити найсвіжішу інформацію.

• Створювати сховище документів для групи системного адміністрування не так страшно, як може здатися. Вибираючи, що саме документувати, ви керуєте масштабом сховища. Наведені в цьому розділі зразки допоможуть вам в починанні. Технологія Wiki дозволяє документам динамічно зростати і розвиватися.

• Часто буває важко скласти новий документ. Технологія Wiki полегшує завдання розробки нового документа, створюючи для вас все посилання. Початковий документ може бути простою технологічною картою, яка згодом буде доповнюватися. Вам зовсім не обов'язково з самого початку створювати бездоганний документ. Напишіть те, що важливо в даний момент, і дозвольте документу розвиватися.

**Особисте сховище інформації**

Деякі відомості мені хочеться мати під рукою, де б я не знаходився, але пам'яті портативного комп'ютера недостатньо для їх зберігання. Я не прагну до того, щоб ця інформація була доступна завжди і відразу, але бажано мати до неї хоч якийсь доступ. Розміщення в Мережі робить її доступною майже з будь-якого місця, особливо в міру поширення технології WiFi. Встановити каталог, захищений паролем, досить просто.

Частина інформації я тримаю в сховище Subversion. Подібно CVS і Microsoft SourceSafe, технологія Subversion дозволяє мені читати і оновлювати інформацію в сховище з будь-якої точки мережі. Технологія Subversion часто застосовується для зберігання вихідного коду і відстеження змін. Теоретично, де б я не знаходився, я можу або завантажити останню версію сховища, або звернутися через SSH до комп'ютера, у якого є зв'язок зі сховищем. В особистому сховище я тримаю дуже велику адресну книгу і деяку іншу інформацію.

**13.        Автоматизація**

• Автоматизація - велика річ. Вона економить ваш час. Крім того, вона дозволяє вам доручити роботу іншому менш кваліфікованому працівнику.

• Є чотири типи завдань, з якими стикаються системні адміністратори:

• Прості завдання, що виконуються одноразово

• Складні завдання, що виконуються одноразово

• Прості завдання, що виконуються часто

• Складні завдання, що виконуються часто

• Складні одноразові задачі і прості завдання, що виконуються часто, є найбільш підходящими кандидатами на автоматизацію. Незважаючи на можливий спокуса автоматизувати складні завдання, що виконуються часто, для їх виконання краще використовувати готові програмні пакети (комерційні або безкоштовні).

• Щоб автоматизувати процес, спочатку переконайтеся, що ви в змозі виконати його вручну. Потім інвентаризують всі кроки і автоматизуйте кожен крок окремо. Після цього зберіть кроки воєдино.

• Використовуючи псевдоніми, ви полегшите собі введення команд. Це справедливо і щодо систем з командним рядком, і для додатків, наприклад SSH. Встановіть псевдонім по можливості максимально близько до додатка. Наприклад, якщо задати псевдонім в файлі конфігурації SSH, то все з истема, що працюють з SSH, будуть використовувати цей псевдонім.

• Команда make систем UNIX / Linux - виключно потужний інструмент. Вона призначена не тільки для програмістів. Ви можете застосовувати її для автоматизації завдань системного адміністрування. У системах UNIX / Linux, особливо на серверах, створюйте файл Makefile в каталозі / etc для автоматизації повсякденних завдань, таких як індексація псевдонімів, копіювання даних і т. Д.

• Мови оболонок bash і / bin / sh набагато складніше і потужніше, ніж ви думаєте. Приклади, наведені в цьому розділі, демонструють, як слід обробляти опції командного рядка і навіть як написати невеликий детектор шкідливих програм!

• Якщо ви пишете довгу командний рядок, тестируйте її по частинах.

• Якщо ви пишете код для користувачів, зростає важливість інтерфейсу. Існують технічні прийоми, що дозволяють створювати хороші призначені для користувача інтерфейси. Ви можете позбавити себе від турбот, переклавши введення даних на програму на зразок MS Excel. Ви також можете створити програму з меню або веб-інтерфейс, щоб дозволити звичайним користувачам отримувати доступ до привілейованих систем.

• Створюючи код, що дозволяє користувачам виконувати привілейовані операції, будьте гранично уважні. За базу використовуйте наявні добре зарекомендували себе засоби безпеки, такі як sudo. За допомогою таблиці дозволів поясніть керівництву, який користувач які привілеї матиме. Дозволяти або забороняти - справа керівництва, а ваша задача полягає в тому, щоб допомогти йому розібратися в питанні. Перш ніж ввести систему в експлуатацію, заручитися згодою керівництва.

• Розвиваючись як системний адміністратор, ви будете автоматизувати все більше і більше завдань. Я раджу вам вивчити мову програмування, придатний цілей системного адміністрування, наприклад Perl, Python, Ruby або Shell. Також звертатись до організаторів увагу на техніку роботи в конкретних операційних системах. Ці питання висвітлюються в книгах серії «Cookbook» (Збірник рецептів) видавництва O'Reilly, згаданих раніше в цій книзі.

**14 .Епілог**

Вітаю! Ви пройшли весь шлях до кінця. Що тепер?

По-перше, я рекомендую вам перечитати цю книгу. Повтореньемать навчання. Якщо ви перечитаєте (або хоча б перегляньте) книгу, поки вона ще свіжа у вашій пам'яті, позитивний ефект посилиться.

По-друге, ми краще вчимося на практиці. Чим більше ви застосовуєте на практиці прийоми, викладені в цій книзі, тим краще ви ними опановуєте. У якийсь момент ви виявите, що засвоїли ці прийоми досить добре, щоб творчо розвивати їх в залежності від вашого стилю життя або конкретної ситуації. Один мій читач помітив, що йому стало легше вести списки щоденних справ, коли він в своєму записнику поміняв місцями сторінки списку справ і розкладу. Хто б міг подумати! Добре все, що приносить хороші результати! Я лише раджу випробувати спершу мій підхід, щоб отримати уявлення про систему в цілому.

По-третє, будьте готові до пробуксовки. Іноді ви будете повертатися до своїх старих звичок. У цьому немає нічого страшного, якщо ви віддасте собі звіт в тому, що відбувається і при першій нагоді повернетеся до викладеної системі. Не виключено, що вам буде легше це зробити, перечитавши відповідну главу.

По-четверте, ви, ймовірно, захочете прочитати традиційні книги по тайм-менеджменту, не орієнтовані саме на системних адміністраторів. Моя книга орієнтована на специфіку професії системного адміністратора і не приділяє увагу багатьом загальних питань, добре освітленим в інших роботах. Рекомендую прочитати книгу Девіда Аллена (David Allen) «Getting Things Done: The Art of Stress-Free Productivity» 1, [http://www.davidco.com](https://translate.google.com/translate?hl=uk&prev=_t&sl=ru&tl=uk&u=http://www.davidco.com) .

**Що робити з усім цим «новим» вільним часом?**

Прийоми, описані в цій книзі, подарують вам кілька вільних годин (якщо не днів) в тиждень. Економлячи трохи більше години кожен день, ви виконаєте той же обсяг роботи за 4-денний робочий тиждень.

Що робити з цим вільним часом?

Будь ласка, не витрачайте його даремно. Благаю вас. Щойно розпочавши втілювати прийоми тайм-менеджменту в своє життя, я став віддавати вивільнився моєї тодішньої пристрасті - читання новин Usenet. Думаю, сучасний еквівалент цього заняття - читання RSSканалов з новинами, блогів, веб-сайтів і т. Д. Все це марна трата часу. Будь ласка, не витрачайте свій знову знайдене час, множачи непродуктивні справи в своєму житті.

У мене є ідея трохи краще.

Відведіть це знову знайдене час на боротьбу з несправедливістю.

Найпоширеніший вид несправедливості, щодня спостерігається мною, - це те, як корпорації крадуть наші життя у наших сімей. Одного ранку ми прокидаємося і виявляємо, що наші діти виросли, не знаючи нас, або що наші улюблені пішли, тому що нас майже ніколи не було поруч. І ми ще дивуємося: «Як швидко промчало час!»

Раніше існував так званий «неявний суспільний договір». Ми працюємо на фірму 40 годин в тиждень, а вона платить нам, щоб у нас були гроші на прожиття, і щось відраховує до пенсійного фонду нам на старість. Це була чесна угода. Але тепер корпорації відбирають у нас все більше часу без якої б то не було компенсації. Професіонал працює 60-70 годин на тиждень, а потім підпадає під масове скорочення штатів через помилкових рішень безглуздих топ-менеджерів, зарплата яких в 100, а то й в 1000, раз перевищує його зарплату. У 1990-ті роки я працював в AT & T / Lucent, і нам постійно нагадували, що можуть звільнити нас в будь-який момент незалежно від того, наскільки добре ми справляємося з роботою. нам говорили, що треба радіти переходу від гарантованих пенсій до принципу «кожен сам за себе» відповідно до нового пенсійним планом, прийнятим фірмою. І при цьому в останні роки моєї роботи в цій компанії керівництво було уражено і стривожено зниженням лояльності співробітників. Лояльність - вулиця з двостороннім рухом.