ЛЕКЦІЯ 11. ВЗАЄМОДІЯ З КОРИСТУВАЧЕМ В ОПЕРАЦІЙНИХ СИСТЕМАХ

План

17.1. Термінальне введення-виведення
17.2. Командний інтерфейс користувача
17.3. Графічний інтерфейс користувача
17.4. Процеси без взаємодії із користувачем

17.1. Термінальне введення-виведення

У цьому розділі зупинимося на базовій технології для організації взаємодії із користувачем в операційних системах — термінальному введенні-виведенні.

17.1.1. Організація термінального введення-виведення

Спочатку розглянемо принципи організації термінального введення-виведення, що не залежать від конкретної ОС.

Поняття термінала

Історично *термінали* (terminals) використовували для організації багатокористувацької роботи із мейнфреймами або мінікомп'ютерами. Це були апаратні пристрої, що складалися із клавіатури і дисплея, які підключали до комп'ютера через інтерфейс послідовного порту.

Емуляція термінала

У сучасних умовах апаратні термінали застосовують рідко. Проте інтерфейс зв'язку з терміналами (термінальне введення-виведення) не втратив свого значення й досі. Це пояснюється тим, що текстовий режим роботи дуже зручний для розв'язання багатьох задач (організації адміністрування системи, віддаленого доступу до неї тощо).

Для спрощення організації термінального введення-виведення у сучасних ОС широко використовують *емуляцію термінала.* Програмне забезпечення (емуляор) приймає дані згідно із домовленостями щодо обміну із відповідним терміналом і відображає на дисплеї комп'ютера інформацію згідно керуючих послідовностей, визначених для термінала (для цього може бути виділене окреме вікно на графічному екрані).

Віддалені термінали і консоль

Емулятор термінала може бути запущений на віддаленому комп'ютері, при цьому необхідно забезпечити обмін даними мережею між ним і програмним забезпеченням. Прикладом розв'язання такої задачі є протокол telnet, який працює поверх TCP/IP. Відповідний сервер (telnet-сервер) запускають на машині, яка надає віддалений доступ. Він перехоплює дані, що їх застосування передають на термінал, і пересилає їх на віддалену систему. Там працює емулятор термінала (telnet-клієнт), який інтерпретує отримані дані й, у свою чергу, відсилає серверу інформацію, введену на віддаленій машині. Сервер доставляє її застосуванням.

З іншого боку, якщо дисплей комп'ютера, на якому запущена ОС, працює у текстовому режимі, за відображення інформації на ньому теж може відповідати емулятор термінала. У багатокористувацьких ОС із мережним доступом (наприклад, в UNIX) такий термінал часто називають *консоллю,* на відміну від терміналів, які використовують для доступу до системи через мережу.

Термінальне введення

Є два підходи до організації термінального введення.

♦ У ***режимі без обробки,*** або ***неканонічному режимі*** (non-canonical mode), дані передають програмі без зміни (включаючи керуючі коди, такі, як переведення каретки або Backspace). За інтерпретацію цих кодів відповідає програма. Такий режим складніший у використанні (потрібно інтерпретувати керуючі коди), але більш гнучкий. Найчастіше його використовують текстові редактори.

♦ У разі використання ***режиму з обробкою,*** або ***канонічного режиму*** (canonical mode), дані додатково оброблятимуться перед тим як надійти у програму. Така обробка відбувається після натискання користувачем клавіші Enter (введення символу переведення рядка), при цьому керуючі коди буде інтерпретовано і відповідно до них змінено весь уведений рядок (наприклад, якщо в ньому тричі поспіль трапиться Backspace, ці три символи і ще три, введені перед ними, із рядка будуть вилучені). Такий режим простіший для програміста, у програму в даному разі потрапляє вже підготовлений символьний рядок.

Прикладом програмного забезпечення, що реалізує компроміс між цими режимами, може бути розповсюджена в UNIX-системах бібліотека readline. Вона надає розширені засоби редагування введеного рядка, які потребують підтримки неканонічного режиму, але її програмний інтерфейс аналогічний до введення в канонічному режимі (у програму потрапляє підготовлений внаслідок редагування рядок).

Введені із клавіатури символи зберігаються у буфері, навіть у неканонічному режимі (трапляються ситуації, коли застосування не може відразу прийняти дані від клавіатури, і потрібно зберегти їх до того моменту, коли з'явиться така можливість). У канонічному режимі дані із буфера передаються програмі після введення символу переведення рядка, у неканонічному — як тільки програма буде готова їх прийняти. Звичайно для кожного термінала створюється свій окремий буфер введення. У разі заповнення буфера може бути виділена додаткова пам'ять.

Термінальне виведення

Для виведення на термінал теж використовують буферизацію. Буфер виводу заповнюють у тому випадку, коли термінал не готовий прийняти символ; у міру його готовності символи із буфера передають терміналу. Відображаючи дані, він інтерпретує керуючі послідовності, після чого показує інформацію, виділяє кольорами окремі ділянки, переміщає курсор тощо.

Головна проблема полягає в тому, що різні модифікації терміналів сприймають різні набори послідовностей. Для її вирішення у сучасних ОС звичайно створюють базу даних терміналів, що містить список терміналів і послідовності, які відповідають кожному із них. В UNIX-системах таку базу називають termi nfo.

17.1.2. Термінальне введення-виведення в UNIX та Linux

Тут розглянемо особливості реалізації та використання термінального введення-виведення в UNIX-системах на прикладі Linux.

Файли термінальних пристроїв і консоль

Кожному терміналу в Linux відповідає файл символьного пристрою. Наприклад, файли /dev/ttyn *(n=* 1,2,63) відповідають терміналам віртуальної консолі (доступний набір таких терміналів, що дає можливість відкривати кілька паралельних сесій користувача; для перемикання між віртуальними консолями використовують комбінації клавіш Ctrl+Fw), файли /dev/ttySw -терміналам, пов'язаним із з'єднаннями через послідовний порт. Відкривши такий файл, можна працювати із відповідним терміналом.

tty2 = open(/dev/tty2". 0_RDWR. 0644):

write(tty2. "Виведення на другу віртуальну консоль\n". ...):

Консоль Linux емулює спеціальний вид термінала, який називають Linux. Він надає доволі широкі можливості щодо керування відображенням інформації (підтримку кольору, керуючих клавіш, перевизначення символьної таблиці «на ходу»). Поточну консоль відображають файлом /dev/console.

Керуючий термінал процесу

Процес в UNIX-системі може мати *керуючий термінал* (controlling terminal), з якого отримуватиме сигнали від клавіатури (SIGINT у разі натискання користувачем Ctrl+C, SIGQUIT — Ctrl-D). Звичайно це термінал, із якого ввійшов у систему користувач, що створив такий процес. Для процесу доступний файл /dev/tty, що відповідає цьому терміналу. Далі в розділі ознайомимося із деякими додатковими особливостями взаємодії між процесами і керуючими терміналами.

17.1.3. Термінальне введення-виведення у Win32 API

Основним для термінального введення-виведення у Win32 є поняття *консолі*. Воно відрізняється від визначеного раніше; фактично консоль — це наданий ОС спеціальний емулятор термінала.

Звичайно консоль пов'язують із конкретним процесом. Для процесів, які запускає ОС, консолі пов'язують із *консольними процесами,* точкою входу для яких є функція main(). Під час виклику CreateProcess() виділення окремої консолі для процесу задається вмиканням прапорця створення CREATE_NEW_CONSOLE. Крім того, кілька процесів можуть спільно використовувати одну й ту саму консоль (наприклад, після виклику CreateProcess() новий процес за замовчуванням успадковує консоль предка).

Для роботи із консоллю є два набори функцій [1]. Функцй високого рівня дають змогу працювати зі стандартними вводом та виводом і визначати деякі режими керування консоллю. Функції низького рівня дають можливість застосуванням отримувати повну інформацію про інтерактивну роботу користувача із клавіатурою і мишею. У більшості випадків застосуванню достатньо функцій високого рівня; прикладом застосування, розробленого із використанням функцій низького рівня, є файловий менеджер far

17.2. Командний інтерфейс користувача

17.2.1. Принципи роботи командного інтерпретатора

Основним завданням командного інтерпретатора є підтримка інтерактивної роботи користувача, що взаємодіє із системою через термінал.

В UNIX-системах командний інтерпретатор називають *оболонкою* (shell). Розроблено багато версій інтерпретаторів, серед них sh (вихідний варіант), csh (C-shell) і bash. Інтерпретатор bash входить у стандартну поставку більшості дистрибутивів Linux. Системи лінії Windows XP включають спеціалізований інтерпретатор cmd, який використовують під час роботи в режимі консолі. Роботу інтерпретатора буде розглянуто на прикладі bash.

Командний інтерпретатор запускають щоразу, коли користувач реєструється у системі із термінала, при цьому стандартним вхідним і вихідним пристроєм для інтерпретатора і запущених за його допомогою програм є цей термінал. Під час запуску зчитують конфігураційні файли і виконують визначені в них дії із підготовки середовища для цього користувача.

Під час роботи інтерпретатор очікує на введення даних користувача, відображаючи підказку (наприклад, знак долара). Після отримання даних користувача (які формують командний рядок) він інтерпретує їх і виконує деякі дії. Найчастіше вони зводяться до виконання програми, для чого інтерпретатор створює процес, завантажує в нього програмний код і очікує його завершення (відповідно до технології fork+exec). Наведемо приклад.

$ cat myfi1e.txt вміст файла myfile.txt *$ ... очікування введення*

Унаслідок виконання цього командного рядка буде створено новий процес, куди буде завантажено код утиліти cat, параметром якої є ім'я файла. Утиліта зчитує цей файл і відображає його на стандартний вивід. Після завершення виконання утиліти інтерпретатор подає підказку і очікує введення наступного командного рядка.

Процес може бути запущений асинхронно, для чого наприкінці командного рядка потрібно задати символ &. Після цього підказка видається негайно, а процес продовжує своє виконання у фоновому режимі.

$ updatedb &

$ … очікування введення, updatedb продовжує виконання

Тут утиліта updatedb обновлює базу, що містить імена всіх файлів на файловій системі (для наступного пошуку потрібних імен за допомогою утиліти "locate), що є тривалою операцією, яка не потребує втручання користувача. Внаслідок запуску ця утиліта починає виконання у фоновому режимі, а користувач може відразу вводити нові команди.

Набори команд інтерпретатора можуть зберігатися в командних файлах (такі інтерпретатори, як bash, дають можливість використати в них досить потужну мову програмування). Цей командний файл може бути виконаний за тими самими правилами, що і будь-який файл скрипта.

17.2.2. Переспрямування потоків введення-виведення

Важливою технологією, яку реалізують командні інтерпретатори, є переспрямування потоків введення-виведення. При цьому програма замість використання термінала для стандартного потоку введення і стандартного потоку виведення працює із файлом.

$ sort > out.txt $ sort < in.txt

Під час виконання таких команд результат виведення опиниться не на екрані, а у файлі out.txt, а введення буде виконано не з клавіатури, а з файла in.txt. При цьому очікування введення із клавіатури не буде.

Можна переспрямовувати одночасно і потік введення, і потік виведення:

$ sort < in.txt > out.txt

У даному випадку програма зчитує дані з одного файла, обробляє їх і записує в інший файл.

Стандартний дескриптор, що відповідає файлу повідомлень про помилки stderr, при цьому не переспрямовують. Для його переспрямування використовують такий синтаксис:

$ sort 2> err.txt

Переспрямування потоків введення-виведення у Win32

Для реалізації переспрямування потоків введення-виведення в рамках одного процесу у Win32 використовують функцію SetStdHandle() BOOL SetStdHandle(DWORD std_const, HANDLE fh);

Тут std_const набуває тих самих значень, що і для GetStdHandl e(), a fh відображає відкритий файл.

DWORD bytes_written:
HANDLE fd. stdout:
// одержати стандартний дескриптор
stdout = **GetStdHandle**(STD_OUTPUT_HANDLE):
// вивести дані у файл стандартного виводу
WriteFile(stdout. "на консоль\п". 11, &bytes_written. 0);
fh = CreateFileCoutput-log.txt", GENERIC_WRITE.);
// переспрямування виведення, аналог royprog.exe > output-log.txt
SetStdHandle(STD_OUTPUT_HANDLE. fh);

WriteFileCstdout, "у файл\n". 7. &bytes_written, 0);

17.2.3. Використання каналів

Кілька фільтрів можна об'єднувати для обробки даних за допомогою каналів, при цьому стандартний вивід одного процесу переспрямовують на стандартний ввід іншого:

$ grep linux * | sort

У цьому разі результати виконання утиліти grep передаватимуться на стандартний ввід утиліті sort. Канали можуть об'єднувати будь-яку кількість процесів.

Ще один спосіб використання каналів зводиться до обміну даними між процесом і командним інтерпретатором, під час якого результати виконання процесу будуть підставлені в командний рядок інтерпретатора. Таку технологію називають командною підстановкою (command substitution), для цього командний рядок виклику застосування, результати виконання якого потрібно використати, беруть у зворотні лапки: 'виконуваний_файл параметри'.

$ grep linux "cat filelist.txt'

У даному випадку файл filelist.txt містить список імен файлів, у яких потрібно зробити пошук. Внаслідок виконання утиліти cat список має видаватись на стандартний вивід, але замість цього його підставляють у командний рядок виклику утиліти grep.

17.3. Графічний інтерфейс користувача

Засоби організації графічного інтерфейсу користувача неоднакові для різних ОС. Спільним у них є набір основних елементів реалізації, куди входять вікна з елементами керування (кнопками, смугами прокручування тощо), меню і піктограми, а також використання пристрою для переміщення курсору по екрану та вибору окремих елементів (наприклад, миші).

Розрізняють два основних підходи до реалізації графічного інтерфейсу користувача. Для першого характерна тісна інтеграція у систему засобів його підтримки (вони, наприклад, можуть бути реалізовані в режимі ядра). Другий реалізує підтримку такого інтерфейсу із використанням набору застосувань і бібліотек рівня користувача, який ґрунтується на засобах підсистеми введення-виведення. У цьому розділі як приклад інтегрованої підтримки графічного інтерфейсу користувача буде описано віконну і графічну підсистеми Windows XP, а як приклад реалізації його підтримки в режимі користувача — систему X Window.

Інтерфейс віконної та графічної підсистеми Windows XP

Базовим елементом Win32-застосування, яке використовує можливості віконної та графічної підсистеми, є вікно, де це застосування може відображати свою інформацію. Кожне вікно належить деякому *класу вікна* (window class), який реєструється у системі. Програма починає своє

виконання із реєстрації класу вікна, після цього об'єкт вікна може бути розміщений у пам'яті та відображений.

Взаємодія із користувачем у застосуваннях, що використовують вікна, відрізняється від тієї, про яку йшлося під час розгляду консольних застосувань. Основним тут є те, що застосування має бути завжди готовим виконати код у відповідь на дії користувача. Наприклад, у будь-який момент може знадобитися перемалювати зображення внаслідок того, що користувач змінив розмір вікна або зробив його активним, вибравши відповідну піктограму мишею на лінійці задач.

Таку постійну готовність складно реалізувати із використанням послідовного виконання коду. У Win32 (і більшості систем, що реалізують графічний інтерфейс користувача) використовують інший підхід до організації програмного коду: в ній реалізовані *застосування, керовані повідомленнями* (message-driven applications). Усі дії користувача (введення із клавіатури, переміщення миші тощо) ОС перехоплює і перетворює у *повідомлення* (messages), які скеровує застосуванню, що володіє вікном, із яким працював користувач. Код застосування містить *цикл обробки повідомлень,* де відбуваються очікування повідомлень та їхні необхідні перетворення, а також оброблювач повідомлень, що його викликають у разі отримання кожного повідомлення. В оброблювачі реалізовано код, який визначає реакцію застосування на ту чи іншу дію користувача. Цикл обробки повідомлень триває доти, поки в нього не потрапить особливе повідомлення, що змушує завершити роботу застосування.

17.4. Процеси без взаємодії із користувачем

Дотепер ми розглядали особливості організації взаємодії із користувачем під час розробки інтерактивних процесів. Ще одним видом такої організації є відсутність взаємодії, що становить основну характеристику фонових процесів. Особливості їхньої розробки є темою цього розділу.

17.4.1. Фонові процеси на основі POSIX

У цьому розділі ознайомимося із особливостями розробки фонових процесів у UNIX-сумісних системах. У них фонові процеси за традицією називають *демонами* (daemons). До стандартних демонів Linux належать: init, що є предком усіх процесів системи; cron, що забезпечує запуск програм у певні моменти часу; sendmail, що забезпечує відсилання та отримання електронної пошти тощо.

Сесії та групи процесів

Кожен процес належить до групи процесів, яку позначають ідентифікатором групи процесів (pgid). Ядро може виконувати певні дії над усіма процесами групи (наприклад, відсилати їм усім сигнал). Кожна група може мати лідера групи (у цього процесу ідентифікатор (pid) збігається з ідентифікатором групи). Звичайно процес успадковує

ідентифікатор групи від свого предка, він може також явно змінити свою групу системним викликом setpgrp().

Усі процеси групи володіють одним і тим самим керуючим терміналом у конкретний момент часу, під час виконання він може змінюватися. У групи такого термінала може і зовсім не бути — у цьому разі її процеси не отримуватимуть сигналів від клавіатури.

Кілька груп процесів об'єднують у *сесію.* Для сесії задають ідентифікатор сесії (sid). Звичайно процеси сесії відповідають набору процесів, які створив користувач під час інтерактивного сеансу роботи з системою.

Кожний керуючий термінал пов'язаний із сесією та з однією групою процесів цієї сесії (таку групу називають ще *інтерактивною* (foreground) групою, усі інші називають *фоновими* (background), з ними термінали не пов'язані). Протилежне, як і для груп, не завжди вірне — сесія може бути взагалі не пов'язана із керуючим терміналом, у цьому випадку в неї відсутня інтерактивна група.

У кожної сесії є лідер сесії, який відповідає за керування сесією та ізоляцію її від інших; його ідентифікатор збігається з ідентифікатором сесії. Якщо лідер сесії пов'язаний із терміналом, у разі виходу користувача з системи цей процес отримує сигнал SIGHUP. Звичайною реакцією на цей сигнал буде завершення роботи процесу-лідера та всіх процесів його сесії.

Коли лідер сесії із терміналом не пов'язаний, сигналу про вихід він не отримає і зможе продовжити виконання після закінчення сеансу роботи. Це особливо важливо для фонових процесів.

17.4.2. Служби Windows XP

Аналогом демонів у Windows XP є *служби* (services) — фонові процеси, які можуть виконуватися навіть тоді, коли із системою не працює жоден користувач. За керування службами відповідає менеджер служб (Service Control Manager). Він приймає керуючі команди від застосувань і відповідно до них виконує дії зі службами (наприклад, запускає на виконання або зупиняє).

Користувацький інтерфейс менеджера служб реалізовано двома способами:

- за допомогою вікна керування службами (Services), яке викликають через підменю Administrative Tools головного меню системи (це вікно відображає список служб, дає змогу запускати і зупиняти окремі служби, дізнаватися про їхні властивості тощо);

- за допомогою утиліти net.exe, що входить у поставку Windows XP; наприклад, команда net start ім'я_служби дає команду менеджерові запустити відповідну службу, net stop і м'я_служби - зупинити її.

Для керування службами необхідно мати адміністративні права у системі.

Висновки

♦ Термінальне введення-виведення реалізує взаємодію з алфавітно-цифровими пристроями. У сучасних ОС такі пристрої найчастіше є емуляторами термінала, роботу із якими здійснюють за одними й тими самими правилами. На основі термінального введення-виведення реалізують командний інтерфейс користувача ОС у вигляді командних інтерпретаторів.

♦ Є різні підходи до організації графічного інтерфейсу користувача, найпоширенішим із них є реалізація такого інтерфейсу як інтегрованої частини системи, що працює в режимі ядра (так зроблено у системах лінії Windows XP), і реалізація засобів його підтримки в режимі користувача у вигляді набору бібліотек та утиліт (прикладом є система X Window).

♦ Розробка фонових застосувань, що не взаємодіють із користувачем, здійснюється за особливими правилами. В UNIX-системах для таких застосувань закрито можливість інтерактивного обміну даними із користувачем, у системах лінії Windows XP є спеціальний компонент, відповідальний за керування ними.

Контрольні запитання та завдання

1. Типи терміналів та їхнє призначення.
2. Підходи до організації термінального введення та різниця між ними.
3. Застосування консолі в ОС
4. Командний інтерпретатор Linux
5. Організація циклу обробки повідомлень при розробці графічного інтерфейсу користувача у Windows
6. Основні фонові процеси в UNIX-сумісних системах
7. Призначення служб Windows XP