Phát triển PMMNM

Đặt vấn đề

- PMMNM là gì? (FOSS-free opensource sofware)
- Sự khác nhau giữa PMMNM và PM bản quyền ở chỗ nào
- Tại sao lại lựa chọn MNM
- Những thao tác thường gặp khi viết một PM?

Tại sao lại lựa chọn MNM

- Lý do lựa chọn FOSS là
  - Chứa các tiêu chuẩn mở
  - Mang tính định tính, định lượng, đổi mới
  - Tự do chọn lựa
  - Mang tính linh động cao
  - An toàn

Mục đích của khóa học

- Tìm hiểu về ý nghĩa của MNM
- Hiểu lợi ích của MNM
- Học về cách sử dụng MNM
- Tìm hiểu và phát triển một số PMMNM

Nội dung chính

- Thế nào là FOSS
- Lịch sử của PMMN
- Tìm hiểu cộng đồng phát triển MNM
- Các PMMNM trong kinh doanh và nghiên cứu
- Các công cụ phát triển
- Các ví dụ về xây dựng ứng dụng dựa trên PMMNM
- Tìm hiểu về Hệ điều hành MNM LINUX/UNIX, tìm hiểu NETBEAN_JAVAFX

Những lưu ý chính khi PTPM

- Các nguyên tắc cơ bản của hệ thống máy tính
- Mã nhị phân (Binary code)và mã nguồn(source code)

Tại sao mã nguồn lại quan trọng

- Tìm hiểu bộ biên dịch
- Các ngôn ngữ lập trình

Các nguyên tắc cơ bản của hệ thống máy tính

- Kiến trúc voneuman
  - Thực hiện một chuỗi các chỉ dẫn (lệnh) chứa trong bộ nhớ
  - Ngôn ngữ máy
    - ✓ Binary code
    - ✓ Khó cho người dùng tìm hiểu
    - ✓ Phát triển từ 8 bit lên 16 bit, 32 bit và giờ là 64 bit

Binary code và Source code

- Binary code
- ✓ Mã máy
- Ví dụ: là một tập hợp các chỉ dẫn được thực hiện trực tiếp bởi CPU
- Được mô tả bằng số hệ 16
- Byte code
- ✓ Được thực thi bởi máy ảo (virtual machine)
- ✓ Ví dụ : Dùng cho JAVA

- Source code
- Các ngôn ngữ lập trình
  - ✓ Dễ hiểu với mọi người
  - ✓ Có thể sửa đổi
- Yêu cầu phải chuyển sang mã nhị phân
  - ✓ Chuyển đổi bằng bộ biên dịch

Các kỹ năng chính cho xây dựng MNM

- Các nguyên tắc cơ bản của MNM
- ✓ Hiểu MNM
- ✓ Các kiến thức liên quan đến phát triển PMMNM
- Tìm hiểu hệ thống UNIX
- ✓ Các thao tác trong UNIX
- ✓ Quản lý hệ thống UNIX
- ✓ Quản lý serve UNIX

- Môi trường PT PMMNM
- ✓ Các công cụ PT PMMNM
- ✓ Các thành phần của một PMMNM
- Công nghệ

Lựa chọn các công nghệ khi cần bao gồm:

- Databases (T1)
- Networks (T2)
- Web services (T3)
- Middle-ware (T4)
- Multimedia (T5)

Khái niệm PMNM

- Định nghĩa (David Wheeler)

  Chương trình phần mềm nguồn mở là những chương trình mà quy trình cấp phép sẽ cho người dùng quyền tự do chạy chương trình theo bất kỳ mục đích nào, quyền nghiên cứu và sửa đổi chương trình, quyền sao chép và tái phát hành phần mềm gốc hoặc phần mềm đã sửa đổi (mà không phải trả tiền bản quyền cho những ngườI lập trình trước)

Các học thuyết về PMNM

Hai học thuyết PMNM chủ đạo

- *Tổ chức phần mềm tự do FSF (Free Software Foundation)*
- *Chương trình Sáng kiến nguồn mở OSI (Open Source Initiative)*

Các học thuyết về PMNM (tt)

Học thuyết FSF

Phần mềm miễn phí nhằm mục đích bảo vệ 4 quyền tự do của người dùng

1. Quyền tự do chạy một chương trình với bất kỳ mục đích nào
2. Quyền tự do nghiên cứu cách thức vận hành của một chương trình và thích ứng nó cho phù hợp với nhu cầu của mình.
3. Quyền tự do phân phát các phiên bản của phần mềm để giúp đỡ những người xung quanh
4. Quyền tự do thêm mới các chức năng cho một chương trình và công bố những tính năng mới đó đến công chúng để toàn cộng đồng được hưởng lợi.

Các học thuyết về PMNM (tt)

Học thuyết OSI

Chú trọng giá trị kỹ thuật của việc tạo ra những phần mềm mạnh, có độ tin cậy cao và phù hợp với giới kinh doanh, đặc biệt là lợi ích thực tiễn của phương pháp xây dựng và quảng bá PMNM

Ưu điểm của phương pháp xây dựng PMNM

1. Giảm sự trùng lặp nguồn lực
2. Tiếp thu kế thừa
3. Quản lý chất lượng tốt hơn
4. Giảm chi phí duy trì

Lịch sử của PMNM

Các cột mốc đáng nhớ

1984: Richard Stallman sáng lập dự án GNU (GNU Not Unix)

1991: Linus Tovards viết thành công lõi Linux

1997: GNU/Linux chiếm 25% thị trường máy chủ

1998: Netscape công bố mã nguồn Navigator

Thuật ngữ "Nguồn mở" ra đời

Thành lập Sáng kiến nguồn mở OSI

Lợi ích của PMNM

1. Tính kinh tế
2. Tính an toàn
3. Tính ổn định
4. Sử dụng chuẩn mở
5. Giảm phụ thuộc vào nhập khẩu
6. Phát triển năng lực ngành CNPM địa phương
7. Giảm tình trạng vi phạm bản quyền

Hạn chế của PMNM

1. Thiếu các ứng dụng kinh doanh đặc thù
2. Tính tương hỗ với các phần mềm đóng kém
3. Giao diện người dùng chưa tốt

Các loại giấy phép PMNM

Giấp phép đại chúng GNU (General Public License)

- Người phổ biến một chương trình đã được cấp phép đại chúng phải đồng thời phổ biến luôn cả mã nguồn cho người nhận
- Nếu người phổ biến chương trình đã thực hiện một sửa đổi gì đó cho phần mềm thì những sửa đổi đó cũng phải được cấp phép theo chế độ giấy phép đại chúng
- Người phổ biến chương trình không áp dụng với người nhận bất cứ hạn chế nào không thuộc phạm vi giấy phép đại chúng
- Người nhận một phần mềm đã cấp phép đại chúng sẽ được trao y nguyên mọi quyền như người phổ biến gốc, tức là quyền sao chép, chỉnh

Các loại giấy phép PMNM (tt)

Giấp phép BSD (Berkeley System Distribution)

- Ghi nhận công lao của tác giả đầu tiên làm ra phần mềm bằng cách đưa vào file mã nguồn các thông tin bản quyền gốc
- Người phát hành ban đầu sẽ không chịu trách nhiệm trước pháp luật về bất cứ thiệt hại nào phát sinh do sử dụng những phần mềm nguồn mở đã được chỉnh sửa

Các loại giấy phép PMNM (tt)

	Giấp phép đại chúng	Giấp phép BSD
Phải phổ biến mã nguồn gốc	Có	Không
Phải phổ biến mã nguồn người dùng tạo mới	Có	Không
Mã nguồn tạo mới phải được cấp phép đại chúng	Có	Không

Những dự án PMNM thành công

1. BIND (máy chủ tên miền DNS)
2. Apache (máy chủ Web)
3. Sendmail (máy chủ thư điện tử)
4. Open Office (bộ ứng dụng văn phòng)
5. Hệ điều hành LINUX/UNIX

BIND (DNS Server)

- Các địa chỉ Internet như yahoo.com hay microsoft.com sẽ không thể thực thi nếu không có Domain Name Servers (DNS). Các servers này sẽ chuyển tên người dùng thành các địa chỉ máy tính được số hóa. Nếu không có các servers này, người sử dụng sẽ không thể có các địa chỉ như 202.187.94.12 để sử dụng trên website.
- Server Berkeley Internet Name Domain (BIND) chạy chiếm 95 % của tất cả các loại DNS
- BIND là một chương trình được cấp bản quyền theo kiểu BSD

Apache (Web Server)

- Để nhận và yêu cầu dữ liệu từ web browsers, Apache là một trong những server nổi tiếng trong thế giới (WWW)
- Apache xuất hiện từ tháng 4 - 1996 và trở thành đối thủ cạnh tranh của IIS (Microsoft ‘ s server)

Apache

- Thiết kế linh hoạt theo module
- Ổn định
- Bảo mật
- Tốc độ nhanh
- Đa nền
- Mã nguồn mở (BSD)

MySQL

- Tốc độ rất nhanh
- Mạnh mẽ
- Hỗ trợ CSDL quan hệ
- Đa người dùng, đa tiểu trình
- Giấy phép mã mở (GPL)

PHP

- Hiệu năng cao
- Giao tiếp nhiều CSDL
- Có sẵn nhiều thư viện hỗ trợ Web
- Giá thành thấp
- Dễ học và sử dụng
- Khả chuyển
- Mã nguồn mở (giấy phép PHP)

Sự kết hợp Apache, MySQL, PHP

- Miễn phí
- Hỗ trợ đa nền
- Được thiết kế tối ưu cho nhau
- Mã nguồn mở

Cài đặt và cấu hình AMP

- Windows: Apache, PHP, MySQL (xampp, appserv, foxserv…)
- Môi trường phát triển tích hợp:
  - Zend Studio
  - Macromedia DreamWeaver

Open Office

- Open office dựa trên source code của StarOffice, là một FOSS tương đương với Microsoft Office. Bao gồm các đặc tính xử lý văn bản, bảng tính và trình chiếu
- Một trong những lợi ích đáng kể của sự chuyển dịch giữa môi trường desktop của Windows tới Open Office là có thể đọc hầu hết các tài liệu của Microsoft Office. Điều này khiến Open Office được sử dụng linh hoạt khi chuyển từ Windows sang Linux.

glassfish

- Là PMMNM
- Dùng cho các ứng dụng sever
- Hiện đang có 3 phiên bản
- V1:v2 và v3
- V3:Nhỏ gọn. Modul hóa, nhanh chóng
- Khởi động rất nhanh (<1ms)
- Tổng Modular và kenel (<100K)

Có thể chạy trên mobile và desktop

opensolaris

Python

- Python là ngôn ngữ lập trình cấp cao vì:Mã được biên dịch thành bytecode và được thực thi.
- Python có thể mở rộng trong C và C++.
- Cấu truc của P rất mạnh phù hợp với LTHDT
- Phù hợp với các ứng dụng logic

JAVAFX trên nền NETBEAN

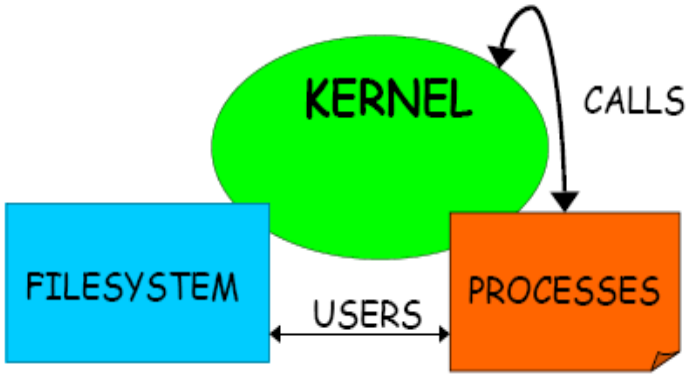
Hệ điều hành Unix/Linux

Hệ điều hành GNU/Linux

1. Được tạo thành bởi sự kết hợp những thành phần trong dự án GNU và lõi Linux
2. Phát hành dưới các bản được đóng gói bởi các Distro: Red Hat, Debian, SuSE, Mandriva…
3. Nguồn mở và miễn phí
4. Phù hợp cho mục đích sử dụng làm máy chủ trong môi trường Internet

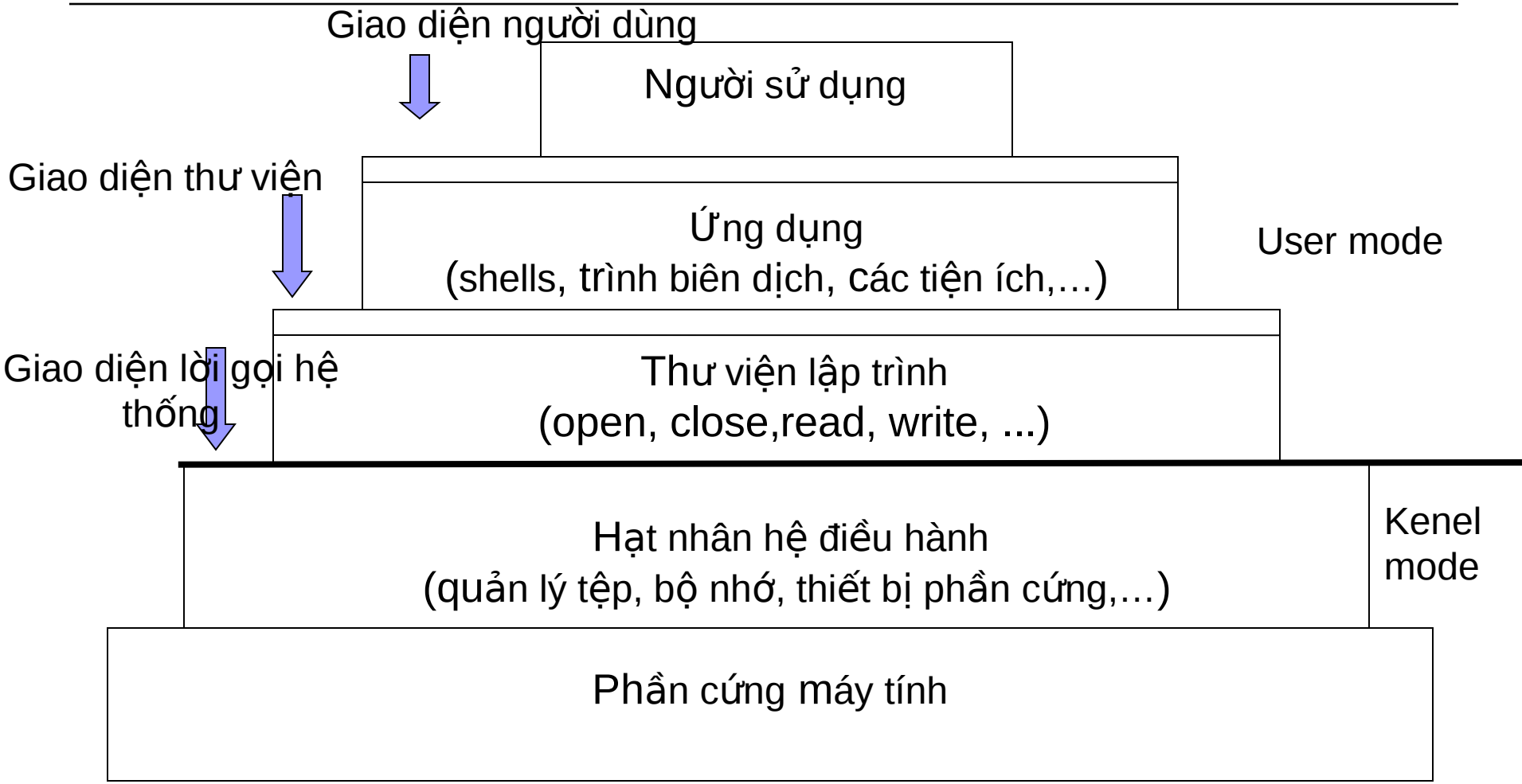
Linux là kernel

- Linux có nguồn gốc từ tên của nhân được tạo bởi Linus Torvalds.
- Một (nhân-kernel) là một điểm trung tâm của HĐH, điều khiển hoạt động của CPU, bộ nhớ và quản lý thiết bị. Nó cũng là môi trường giao tiếp trung gian giữa các chương trình khác nhau chạy trên HĐH.
- Ngoài ra còn có nhân March

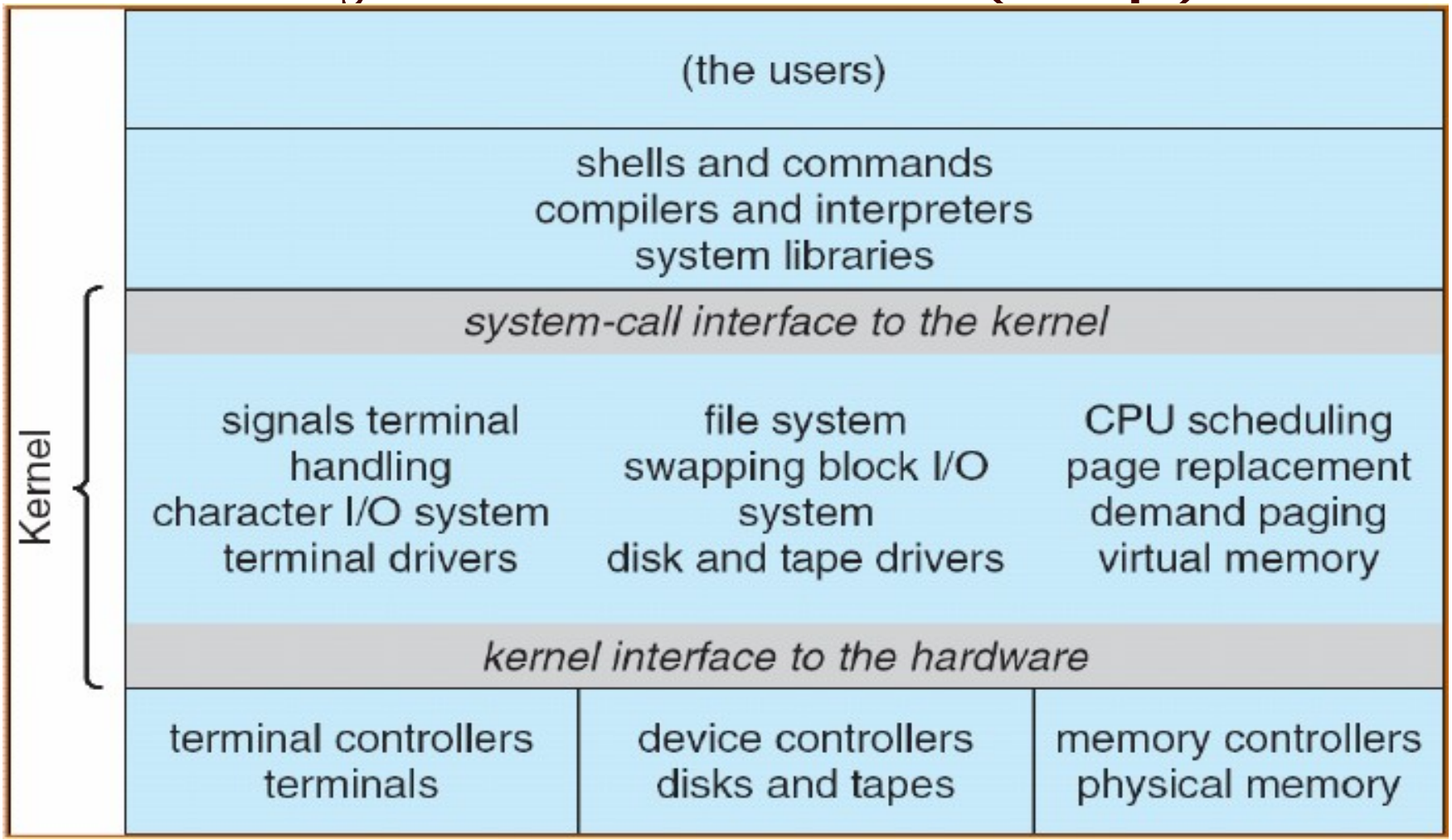


- Hầu hết các ứng dụng FOSS sẽ chạy trên nhân Mach hoặc Linux không quá khó khăn. Tuy nhiên những nhân này thường chịu ảnh hưởng lớn của quá trình thực thi và nền phần cứng mà hệ thống FOSS chạy trên đó.
- Ví dụ Nhân GNU Hurd chỉ có thể chạy trên kiến trúc x86 (PC). Nhân Linux chạy trên hầu hết các kiến trúc máy tính bao gồm Playstation 285, mainframes và các thiết bị nhúng.

Kiến trúc hệ thống



UNIX System Structure (tiếp)



Ý tưởng trong thiết kế Unix/Linux

- Hệ điều hành đa nhiệm (multitasking)
- Hệ điều hành đa người dùng (multiuser)
- Tạo (creation), chỉnh sửa (modification) hay hủy bỏ (destruction):
  - Chương trình
  - Tệp tin.
- File System
- Chia xẻ tài nguyên hệ thống:CPU, RAM,...

Tính đa nhiệm

- Một chương trình khi chạy trong máy tính là một tiến trình
  - Đa nhiệm có nghĩa là nhiều tiến trình có thể chạy cùng một thời điểm
  - Tiến trình không phải là chương trình
  - Có thể chạy nhiều tiến trình cho cùng một chương trình tại một thời điểm

Tính đa người sử dụng

- Nhiều người sử dụng có thể cùng truy xuất vào hệ thống tại một thời điểm
  - cần có khái niệm tài khoản sử dụng nhưng có nhiều tài khoản không đồng nghĩa với đa người sử dụng
  - một tiến trình tạo ra thuộc quyền sở hữu người đã tạo ra nó
  - do đó các tiến trình có thể thuộc quyền sở hữu của nhiều người khác nhau

Tính mô đun

- Mô đun hoá về kiến trúc
  - Hạt nhân quản lý các nhiệm vụ ở mức thấp
  - Tầng ứng dụng cung cấp các tiện ích sử dụng đối với người sử dụng
- Mô đun hoá về ứng dụng
  - Cung cấp nhiều công cụ nhỏ, chuyên dụng nhưng đa dạng để hỗ trợ công việc người sử dụng
  - Không cung cấp các công cụ có tính đa năng nhưng người sử dụng làm được rất nhiều việc phức tạp bằng cách kết

Ý tưởng (tiếp)

- Làm trong môi trường mạng.(Networking)
- Tập chương trình tiện ích đầy đủ. (System Utilities)
- Thư viện lập trình (System call Interface, Standard Library)
- Hệ thống mở, tương thích với nhiều nền phần cứng (platform) khác nhau

Các công cụ cơ bản

- Các trình thông dịch lệnh (shell) : sh, csh, bash
- Các câu lệnh quản lý hệ thống tệp
- Các câu lệnh quản lý tiến trình
- Các câu lệnh xử lý dữ liệu
- Các trình soạn thảo: vi, emacs, …
- Các trình quản lý gói dữ liệu: tar, gzip,…
- Các trình biên dịch : C, C++, Fortran, Perl
- Các bộ xử lý văn bản (latex), hình ảnh (xv)
- v.v.

Lịch sử phát triển

- 1965 – 1969 : MIT + AT&T Bell Lab:
  - Dự án OS Multics (Multiplexed Information and Computing Service)
  - Yêu cầu đặt ra:
    - Hỗ trợ nhiều người dùng cùng làm việc.
    - Hỗ trợ tính toán tốc độ cao,
    - Chia xẻ tài nguyên của người dùng.
  - Dự án không thành công.

Lịch sử phát triển (tiếp)

- 1970 – 1979 :Thomson+ Ritchie Bell Lab:
  - Thiết kế File System.
  - Thiết kế kiến trúc file thực thi trong Unix.
  - Phân chia thời gian (Time sharing).
  - Liên lạc giữa các tiến trình (PIPE).
  - Xử lý văn bản (text processing)→ Shell.
  - Viết lại Kernel bằng C

Lịch sử phát triển (tiếp)

- 1980 – 1989 : Nhiều phiên bản Unix:
  - Bổ sung C Shell và Korn Shell.
  - Xử lý văn bản: PERL và TCL.
  - Hỗ trợ TCP/IP trong Kernel.
  - Xây dựng theo chuẩn POSIX (Portable Operating System Interface).
  - Bổ sung: System Administration, IPC,..
  - Thiết kế giao diện tương tác với người dùng.

Lịch sử phát triển (tiếp)

- 1990 – nay : Unix & Linux:
  - Unix được hoàn thiện hơn về:
    - Kiến trúc Kernel.
    - FileSystem.
    - Tương thích với nhiều platform phần cứng.
  - Chuyển sang Linux:
    - Kernel là mã nguồn mở (free open source).
    - OS Unix-like : Kernel Linux giống Unix (90%).
    - Cộng đồng mã nguồn mở lớn.

Các thành phần của Unix/Linux

- Phân nhân - Kernel

  Chương trình chính điều khiển mọi hoạt động của máy tính

- Phần vỏ - Shell

  Thực hiện giao tiếp với người dùng: thông dịch lệnh người dùng và chuyển tới kernel để thực thi

- Hệ thống tập tin - File System

  Lưu trữ thông tin

  Tổ chức trong cây thư mục

- Các chương trình tiện ích

  Các lệnh tiện ích: soạn thảo văn bản, trao đổi thư điện tử

Các phiên bản Unix

- Hai chuẩn UNIX chính
  - BSD (Berkeley Software Distribution)
  - System V (AT&T Bell Labs)
- Hầu hết các phiên bản khác dựa trên hai chuẩn này
  - Solaris (Sun Microsystems)
  - SCO (Santa Cruz Operation)
  - HP-UX (Hewlett-Packard Unix)

Các phiên bản Linux

- Debian www.debian.org
- Redhat www.redhat.com
- Fedora Core fedora.redhat.com
- SuSe www.suse.com
- Mandrake www.mandrakelinux.com
- SlackWare www.slackware.com
- TurboLinux www.turbolinux.com

Redhat và Fedora Core

- Redhat và Fedora Core Bản Linux có lẽ là thịnh hành nhất trên thế giới, phát hành bởi công ty Redhat.
- Từ 2003, Công ty Redhat phát triển Redhat Enterprise Linux (RHEL) với mục đích thương mại, nhằm vào các công ty, xí nghiệp.
- Redhat đầu tư mở ra dự án Fedora nhằm phát triển phiên bản Fedora Core cho người dùng bình thường.
- Hiện nay đa có Fedora Core 4

Sơ bộ về shell

- Shell (***hệ vỏ***) - bộ dịch lệnh, hoạt động như một kết nối trung gian giữa nhân với người dùng: Shell nhận dòng lệnh do người dùng đưa vào, từ dòng lệnh nói trên, nhân tách ra các bộ phận để nhận được một hay một số lệnh tương ứng với các đoạn văn bản có trong dòng lệnh.Một lệnh bao gồm tên lệnh và tham số.
- Shell sử dụng nhân để khởi sinh một quá trình mới (khởi tạo quá trình) và sau đó, shell chờ đợi QT con này tiến hành, hoàn thiện và kết thúc. Khi shell sẵn sàng tiếp nhận dòng lệnh của người dùng, một dấu nhắc xuất hiện trên màn hình.

- Linux có hai loại shell phổ biến là: C-shell (%), Bourne-shell ($)
- Để nhận biết hệ thống đang làm việc với shell nào, chúng ta gõ lệnh: # **alias**
- Nếu một danh sách xuất hiện là C-shell; ngược lại, nếu xuất hiện thông báo "Command not found" Bourne-shell.

Lệnh

Lệnh chia thành 3 loại:

- Lệnh thường trực (có sẵn của Linux), bao gồm các lệnh chứa sẵn trong shell và các lệnh thường trực khác.
- File chương trình ngôn ngữ máy: người dùng viết chương trình trên ngôn ngữ C qua bộ dịch ***gcc*** để tạo ra một chương trình trên ngôn ngữ máy.
- File chương trình shell (Shell Script).

Chú ý : Khi kết thúc một dòng lệnh cần gõ phím ENTER để shell phân tích và thực hiện lệnh

Cách sử dụng lệnh

- Dạng tổng quát của lệnh Linux

  # <Tên lệnh> [<các tham số>]

- Trong đó

  - Tên lệnh là một dãy ký tự - bắt buộc phải có, không có dấu cách, biểu thị cho một lệnh của Linux hay một chương trình. Người dùng cần HĐH đáp ứng yêu cầu gì của mình thì chọn đúng tên lệnh.
  - Các tham số có thể có hoặc không, nhằm cung cấp thông tin về các đối tượng mà lệnh tác động

- Có hai loại tham số : tham số khóa ("tùy chọn") và tham số vị trí.
  - Tham số khóa chính là những TS điều khiển hoạt động của lệnh. TSK thường bắt đầu bởi dấu trừ "**-**" hoặc hai dấu trừ liên tiếp "**--**"
  - Tham số vị trí là tên file, thư mục & thường là các đối tượng chịu sự tác động của lệnh. Khi gõ lệnh, tham số vị trí được thay bằng những đối tượng mà người dùng cần hướng tác động tới.
- Ví dụ Khi người dùng gõ lệnh xem thông tin về các file:

  # ls -l g*↵

- Trong lệnh này:

ls - tên lệnh liệt kê các tên file/ thư mục con trong một thư mục,

-l - tham số khóa, cho biết yêu cầu xem đầy đủ thông tin về các đối tượng hiện ra. Tương ứng với lệnh **ls** còn có các tham số khóa **-a**, **-L**.

Trong một số tham số khóa có nhiều chữ cái thay cho một dấu "**-**" là hai dấu "**--**" ở đầu tham số.

g* là tham số vị trí chỉ rõ người dùng cần xem thông tin về các file có tên gọi bắt đầu là chữ cái "g".

Chú ý:

- Lệnh phân biệt chữ hoa và thường
- Một số lệnh chỉ người quản trị hệ thống (siêu người dùng) mới được sử dụng
- Một dòng lệnh có thể có nhiều hơn một lệnh, trong đó lệnh sau được ngăn cách bởi với lệnh đi ngay trước bằng dấu ";" hoặc dấu "|". Ví dụ về một số dòng lệnh dạng này:

 # ls -l; date

 # head Filetext | sort >temp

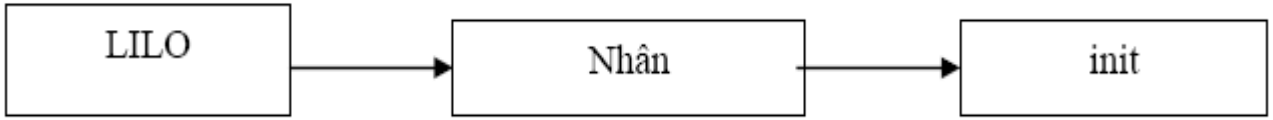
- Sử dụng “man” để tra cứu lệnh

 # man <tên-lệnh>

THAO TÁC VỚI HỆ THỐNG

Quá trình khởi động Linux được mô tả theo sơ đồ sau

- LILO(LInux LOader) được tải vào máy để thực hiện, việc đầu tiên là đưa nhân vào bộ nhớ trong, sau đó tải chương trình init để thực hiện việc khởi động Linux.



- Nếu cài đặt nhiều phiên bản Linux hay cài Linux cùng các HĐH khác (khi đó mỗi phiên bản HĐH được gán ***nhãn*** - label để phân biệt), thông báo sau đây được LILO đưa ra:

LILO boot: cho phép nhập xâu là nhãn của một trong những HĐH hiện có trên máy để khởi động.Ví dụ, gõ

- **LILO boot: linux** nếu chọn khởi động để làm việc trong Linux
- **LILO boot: dos** nếu chọn khởi động để làm việc trong MS-DOS, Windows.

Đăng nhập & thoát khỏi hệ thống

- **Đăng nhập**

  Sau khi hệ thống Linux (lấy Red Hat 6.2 làm ví dụ) khởi động xong, trên màn hình xuất hiện những dòng sau:

  Ret Hat Linux release 6.2 (Zoot) – phiên bản LINUX

  Kernel 2.2.14-5.0 on an i686 – Phiên bản nhân

  May1 login: root

  Password:

  Last login: Fri Oct 27 14:16:09 on tty2

  Root[may1 /root]#

Tên người dùng đăng nhập ***may1*** là tên máy và **/root** tên thư mục hiện thời (người dùng root). Khi dấu nhắc shell xuất hiện trên màn hình - HĐH đã sẵn sàng tiếp nhận một yêu cầu mới của người dùng.

Ra khỏi hệ thống

- **CTRL+ALT+DEL**.
- Sử dụng lệnh **shutdown** với cú pháp như sau: **shutdown [tùy- chọn] <time> [cảnh-báo]**
- Sử dụng lệnh **halt** với cú pháp như sau:

  halt [tùy-chọn] - Lệnh này tắt hẳn máy.
- ***Lưu ý:*** Có thể sử dụng lệnh **exit** để trở về dấu nhắc đăng nhập hoặc kết thúc phiên làm việc bằng lệnh **logout**.
- Khởi động lại:

  reboot [tùy-chọn]

Lệnh xem, thiết đặt ngày, giờ, xem lịch trên hệ thống

- Lệnh xem, thiết đặt ngày, giờ
  - Lệnh **date** cho phép có thể xem hoặc thiết đặt lại ngày giờ trên hệ thống
  - Cú pháp của lệnh gồm hai dạng, dạng xem thông tin về ngày, giờ:

date [tùy-chọn] [+định-dạng]

- Dạng thiết đặt lại ngày giờ cho hệ thống:

date [tùy-chọn] [MMDDhhmm[[CC]YY]].ss]]

Ví dụ…

```
# date
Wed Jan 3 23:58:50 ICT 2001
# date -d='01/01/2000'
Sat Jan 1 00:00:00 ICT 2000
```

Lệnh xem lịch

- Lệnh **cal** cho phép xem lịch trên hệ thống với cú pháp như sau

cal [tùy-chọn] [<tháng> [<năm>]]

Các tùy chọn là:

- -m : Chọn ngày Thứ hai là ngày đầu tuần
- -j : Hiển thị số ngày trong tháng dưới dạng số ngày trong năm
- Hiển thị lịch của năm hiện thời

Ví dụ:

```
# cal 1 2001
```

January 2001

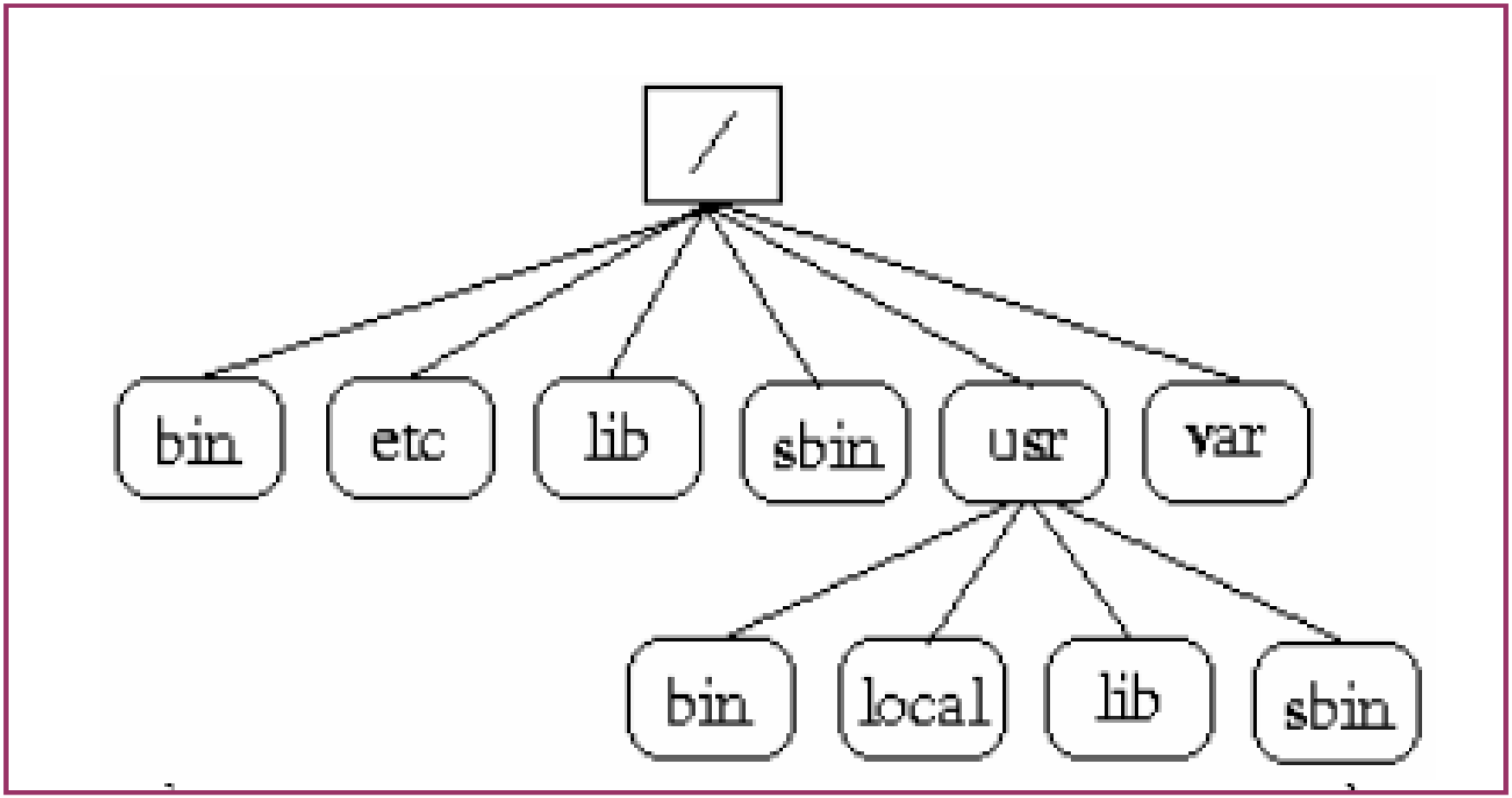
Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

- Dưới đây là ví dụ hiển thị số ngày trong tháng 3 dưới dạng số ngày trong năm 2001

```
# cal -j 3 2001
March 2001
Su     Mo     Tu     We     Th     Fr     Sa
                             60     61     62
63     64     65     66     67     68     69
70     71     72     73     74     75     76
77     78     79     80     81     82     83
84     85     86     87     88     89     90
```

HỆ THỐNG FILE

- Tập hợp tất cả các file có trong HĐH được gọi là ***hệ thống file*** là một hệ thống thống nhất.
- Hệ thống file được tổ chức theo dạng hình cây: Hệ thống file được xuất phát từ một thư mục gốc (kí hiệu "***/***") và cho phép tạo ra thư mục con trong một thư mục bất kỳ.
- Thông thường, khi khởi tạo Linux đã có ngay hệ thống file của nó. Hình 3.1minh họa một phần trong cây lôgic của hệ thống file


/
bin
etc
lib
sbin
usr
var
bin
local
lib
sbin

- Đường dẫn tuyệt đối: Đầy đủ tên thư mục, các thư mục cách nhau bởi “/”

  /etc/X11/xinit/Xclients

- Thư mục hiện thời là thư mục trong hệ thống file mà hiện thời "người dùng đang ở đó".
- Đường dẫn tương đối : Nếu tại thời điểm t, đang ở thư mục xinit, muốn đến **Xclients** ta gõ **Xclients** hoặc **./Xclients** trong đó kí hiệu "**.**" để chỉ thư mục hiện thời.

- Khi một người dùng đăng nhập vào hệ thống, Linux luôn chuyển người dùng vào thư riêng và tại thời điểm đó thư mục riêng là thư mục hiện thời của người dùng.
- TM riêng của siêu người dùng là /root
- TM riêng của người dùng user1 là /home/user1

Quyền truy nhập thư mục và file

- **Quyền truy nhập**

Mỗi file và thư mục trong Linux đều có một chủ sở hữu và một nhóm sở hữu, cũng như một tập hợp các quyền truy nhập.

Quyền truy nhập	Ý nghĩa
---	Không cho phép một quyền truy nhập nào
r--	Chỉ được quyền đọc
r-x	Quyền đọc và thực hiện (cho chương trình và shell script)
rw-	Quyền đọc và ghi
rwx	Cho phép tất cả các quyền truy nhập (cho chương trình)

Các lệnh cơ bản

- **Thay đổi quyền sở hữu file với lệnh** *chown*

  chown [tùy chọn] [chủ][.nhóm] <file ...>

- **Thay đổi quyền truy cập file với lệnh** *chmod*

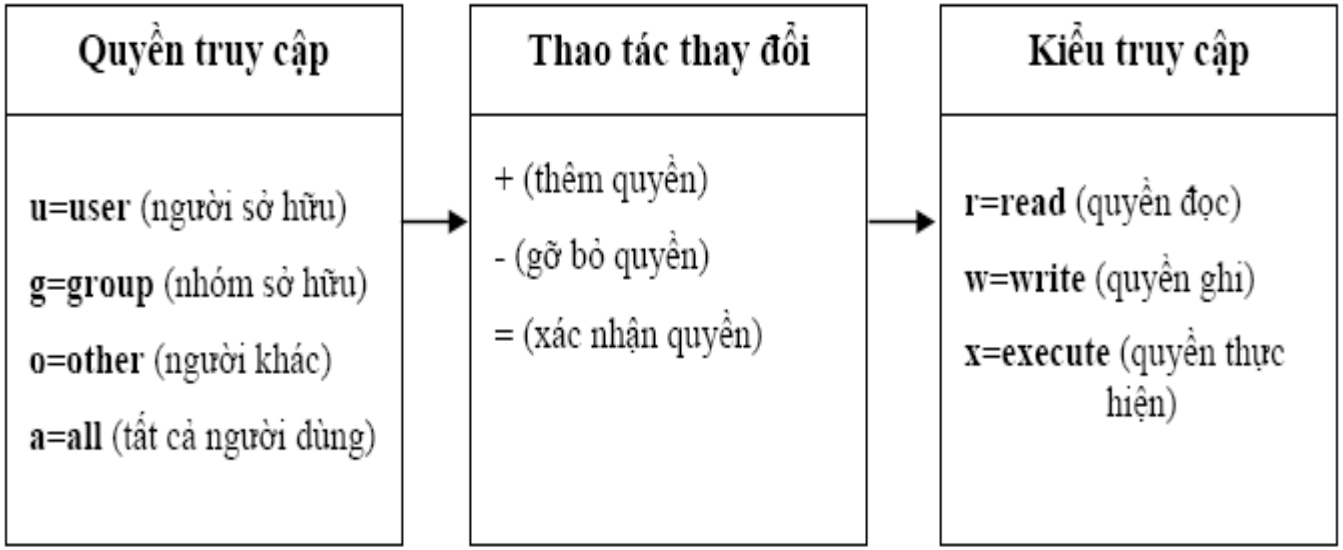
  Cú pháp lệnh **chmod** có ba dạng:

  chmod [tùy-chọn] <mod [,mod]...> <file...> xác lập tương đối

  chmod [tùy-chọn] <mod-hệ-8> <file...> xác lập tuyệt đối

  chmod [tùy-chọn] --reference=nhómR <file...>dạng gián tiếp chỉ dẫn theo quyền truy nhập của file **nhómR**

- Giải thích về hai cách xác lập quyền truy nhập file trong lệnh **chmod** như sau: xác lập tuyệt đối (dùng hệ thống mã số viết theo hệ cơ số 8 biểu diễn cho các quyền truy nhập) và xác lập tương đối (dùng các chữ cái để biểu diễn quyền truy nhập).
- **Cách xác lập tương đối**



- Có thể kết hợp các mục từ hộp thứ nhất và hộp thứ ba với một mục từ hộp thứ hai để tạo ra một **mod**.
- Ví dụ, nếu muốn thêm quyền ghi đối với file ***test*** cho tất cả người dùng trong nhóm sở hữu, hãy chọn **g** cho nhóm sở hữu, **+** cho thêm quyền truy nhập, và **w** cho quyền ghi. Lúc đó lệnh **chmod** sẽ có dạng sau:

  chmod g+w test

- **Cách xác lập tuyệt đối**
- Biểu diễn quyền truy nhập file thông qua dãy gồm 9 vị trí dưới dạng **rwxrwxrwx**, trong đó từng cụm 3 vị trí theo thứ tự tương ứng với: chủ sở hữu, nhóm sở hữu và người dùng khác
- Như vậy, chủ sở hữu tương ứng với 3 bít đầu tiên, nhóm sỡ hữu tương ứng với 3 bít giữa, người dùng khác tương ứng với 3 bít cuối.

- Ví dụ: **chmod 753 memo1** đặt thuộc tính quyền truy nhập đối với file memo1 là **rwxr-xr-x**

Quyền	*Chữ số hệ 8*	*Quyền*	*Chữ số hệ 8*
Chỉ đọc	4	Chỉ đọc và ghi	6
Chỉ ghi	2	Chỉ đọc và thực hiện	5
Chỉ thực hiện	1	Chỉ ghi và thực hiện	3
Không có quyền nào	0	Đọc, ghi và thực hiện	7

Thao tác với thư mục

- **Một số thư mục đặc biệt**

Thư mục gốc */* Đây là thư mục gốc chứa đựng tất cả các thư mục con có trong hệ thống.

Thư mục ***/root*** có thể được coi là "thư mục riêng" của siêu người dùng. Thư mục này được sử dụng để lưu trữ các file tạm thời, nhân Linux và ảnh khởi động, các file nhị phân …

- **Thư mục *bin***
- **Thư mục *dev***
- **Thư mục *dev***
- **Thư mục *lib***
- **Thư mục *lost+found***
- **Thư mục *usr***
- **Thư mục *home***
- **Thư mục *boot*** Là TM chứa nhân của hệ thống

- **Các lệnh cơ bản về thư mục**

  Xác định thư mục hiện thời với lệnh *pwd*

  Cú pháp lệnh: **pwd**

  - **Xem thông tin về thư mục với lệnh** *ls*

    Cú pháp lệnh: **ls [tùy-chọn] [file]...**

  - **Lệnh tạo thư mục *mkdir***_

    mkdir [tùy-chọn] <thư-mục>

  - **Lệnh xóa bỏ thư mục *rmdir***

    rmdir [tùy-chọn[<thư-mục>

- **Lệnh đổi tên thư mục *mv***

  mv <tên-cũ> <tên-mới>

Các lệnh làm việc với file

- **Các lệnh tạo file**

  Tạo file với lệnh *touch*

  touch <file>

- **Tạo file với lệnh *cat***

  cat > <file>

- **Sao chép file với lệnh *cp***

  Lệnh **cp** có hai dạng như sau:

cp [tùy-chọn] <file-nguồn> ... <file-đích>

cp [tùy-chọn] --target-directory=<thư-mục> <file-nguồn>...

- **Đổi tên file với lệnh** *mv*

  Cú pháp lệnh đổi tên file:

  mv <tên-cũ> <tên-mới>

- **Xóa file với lệnh *rm*_**

  Lệnh **rm** rất "nguy hiểm" vì trong Linux không có lệnh khôi phục lại những gì đã xóa.

  Cú pháp lệnh:

- **rm [tùy-chọn] <file> ...**

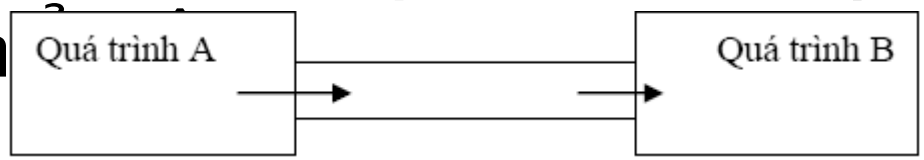
- **Xem nội dung file với lệnh *cat***

  cat [tùy-chọn] <tên file>

LẬP TRÌNH SHELL VÀ LẬP TRÌNH C TRÊN LINUX

- Trong Linux có một số loại shell, shell ngầm định là bash. Shell cho phép người dùng chạy từng lệnh shell (thực hiện trực tiếp) hoặc dãy lệnh shell (file script) và đặc biệt hơn là theo dạng thông qua ống dẫn (pipe).
- Trong một dòng lệnh của shell có thể thực hiện một danh sách các lệnh tuần tự nhau dạng: **<lệnh> [; <lệnh>]...**

- Sử dụng ống dẫn là cách thức đặc biệt trong Linux-một cách thức của shell để truyền thông liên quá trình. ống dẫn được tổ chức theo kiểu cấu trúc dữ liệu dòng xếp hàng "vào trước ra trước" FIFO "First In First Out". Trong cấu trúc dòng xếp hàng, một đầu của dòng nhận phần tử vào và còn đầu kia lại xuất phần tử ra. Trong ngữ cảnh của shell, với hai quá trình A và B được kết nối một ống dẫn được th



Các yếu tố cơ bản để lập trình trong shell

- Shell có công cụ cho phép có thể lập trình trên shell làm tăng thêm độ thân thiện khi giao tiếp với người dùng. Các đối tượng tham gia công cụ như thể có thể được liệt kê:
- Các biến (trong đó chú ý tới các biến chuẩn),
- Các hàm vào - ra
- Các phép toán số học,
- Biểu thức điều kiện,
- Cấu trúc rẽ nhánh,
- Cấu trúc lặp.

- Chương trình là dãy các dòng lệnh shell song được đặt trong một file văn bản (được soạn thảo theo soạn thảo văn bản),
- Các dòng lệnh bắt đầu bằng dấu # chính là chú thích
- Thông thường các bộ dịch lệnh shell là sh (/bin/sh) hoặc ksh (/bin/ksh)
- Để thực hiện một chương trình shell ta có các cách sau đây:

$sh <<tên chương trình>

hoặc **$sh <tên chương trình>**

hoặc nhờ đổi mod của chương trình:

$chmod u+x <tên chương trình>

Chạy chương trình **$<tên chương trình>**

Mục đích

- Hướng dẫn các bước cơ bản sử dụng NetBeans IDE, Micro Edition (Java™ ME platform), ứng dụng Mobile Information Device Profile (MIDP)
- Tìm hiểu về xây dựng ứng dụng mobile bằng công cụ NETBEAN và JAVAME
- Tìm hiểu các bước để xây dựng project Java ME MIDP trên điện thoại mô phỏng.
- Hướng dẫn cách sử dụng các đặc tính IDE để phát triển các ứng dụng CLDC/MIDP

Giới thiệu phần mềm Netbean

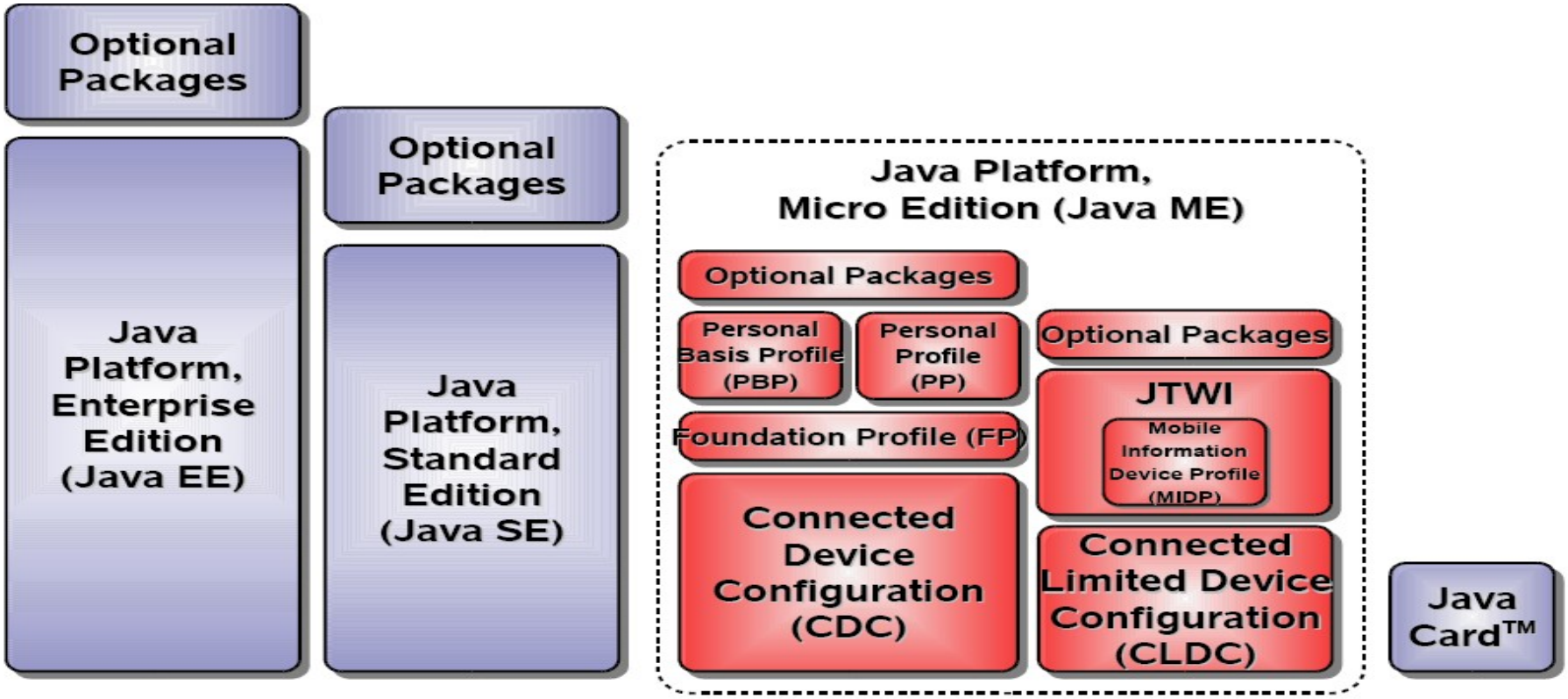
- NetBeans IDE là phần mềm mã nguồn mở - giấy phép GNU (IDE – Integrated Development Environment - Môi trường phát triển tích hợp) cung cấp trình (wizard) cho phép bạn nhanh chóng tạo project MIDP. Khi tạo project bạn có thể chọn phát triển ứng dụng trong môi trường thiết kế Mobile - Visual Mobile Designer (VMD) NetBeans là IDE tốt nhất cho phát triển ứng dụng Java ME
- Sử dụng VMD cho phép bạn thiết kế giao diện đồ họa, thiết kế các màn hình ứng dụng người dùng nhanh, hiệu quả. Các thiết kế đó sẽ được sinh code tự động
- Phát triển các tiện ích web services - server/client
- Giải quyết các vấn đề thiết bị phân mảnh
- Gỡ rối trên các thiết bị thực

Giới thiệu về thiết bị mobile

- Kích thước nhỏ
- Dung lượng bộ nhớ hữu hạn
- Tiêu tốn ít điện năng (thường dùng Pin)
- Bền có thể chịu được va đập khi di chuyển
- Kết nối hạn chế (băng thông thấp)
- Thời gian khởi động ngắn

Tổng quan Java ME

J2ME là một nhánh của ngôn ngữ lập trình JAVA được phát triển nhằm hướng tới việc lập trình cho các thiết bị “nhỏ” (micro) có bộ nhớ, khả năng hiển thị và xử lý hạn chế



Kiến trúc JAVAME

Platform JAVA được phân chia thành các phiên bản khác nhau

J2SE – Java 2Platform, Standard Edition	Các ứng dụng desktop
J2EE – Java 2Platform, Enterprise Edition	Phát triển các ứng dụng web bao gồm servlet, JSP, EJB, XML
J2ME – Java 2Platform, Micro Edition	Ứng dụng mobile và cầm tay
Java Card API	Smart card

Kiến trúc JAVAME

- J2ME - là một tập các công nghệ và đặc tả tập trung vào các thiết bị consumer. Các thiết bị này có dung lượng bộ nhớ hữu hạn, tiêu thụ ít điện, màn hình nhỏ và băng thông mạng thấp
- Java Me cung cấp môi trường linh động cho sự phát triển và chạy ứng dụng trên các thiết bị này
- J2ME giống các chương trình JAVA khác được compile thành byte code và biên dịch bởi Java Virtual Machine (VM)
- J2ME cung cấp giao diện phù hợp với các thiết bị consumer, các ứng dụng này không phải recompiled lại và có thể chạy trên các máy khác nhau

Kiến trúc JAVAME

Nhân của J2ME dựa trên hai khái niệm configuration và profiles

- Một configuration định nghĩa một môi trường runtime cho hệ thống J2ME. Nó định nghĩa nhân các thư viện, máy ảo, các đặc tính mạng và an toàn mạng
- Một profile thêm vào các thư viện cho lớp các thiết bị, cung cấp các giao diện người dùng…

Ngoài ra còn một tập các thư viện tham số hoặc gói các tham số cung cấp một số chức năng thêm vào. Nói chung những gói này trong JAVAME dựa vào dung lượng các thiết bị.

Configuration

Một Configuration định nghĩa các đặc tính tối thiểu cần có của một môi trường runtime Java hoàn chỉnh.

Để đảm bảo tính khả chuyển và các ràng buộc giữa các loại tài nguyên (ràng buộc bộ nhớ, ràng buộc kết nối), các configuration không định nghĩa các đặc tính của các tham số

Một cấu hình J2ME định nghĩa thêm một phần bổ sung tối thiểu của công nghệ JAVA, nó định nghĩa thêm một số thư viện cho một loại thiết bị nào đó

Tóm lại Một Configuration định nghĩa

- Một tập con các ngôn ngữ lập trình JAVA
- Chức năng của máy ảo JAVA (VM)
- Các thư viện lõi
- Các đặc tính mạng và an toàn mạng

Profile

Một profile định nghĩa tập các API và các đặc tính cho các thiết bị. Trong khi configuration định nghĩa các thư viện cơ bản, thì profile định nghĩa các thư viện quan trọng để tạo các ứng dụng hiệu quả, các thư viện này bao gồm giao diện người dùng, mạng và các cách lưu trữ API

CLDC

CLDC - Connected Limited Device Configuarion – Cấu hình thiết bị kết nối hữu hạn định nghĩa như sau:

- Ngôn ngữ JAVA và các đặc tính máy ảo (VM)
- Các thư viện lõi (java.lang.*, java.util.*)
- Input/Ouput (java.io.*)
- An toàn (security)
- Mạng
- Tính quốc tế hóa

Các đặc tính của các thiết bị CLDC

Các thiết bị cài đặt CLDC có các đặc tính sau

- Có bộ nhớ ít nhất 192kb cho nền JAVA (160kb bộ nhớ cho VM và các thư viện, 32 kb bộ nhớ cho runtime VM)
- Bộ xử lý 16 hoặc 32 bit
- Tiêu tốn ít điện (Thường dùng Pin)
- Kết nối mạng với băng thông hữu hạn (thường dùng wireless)

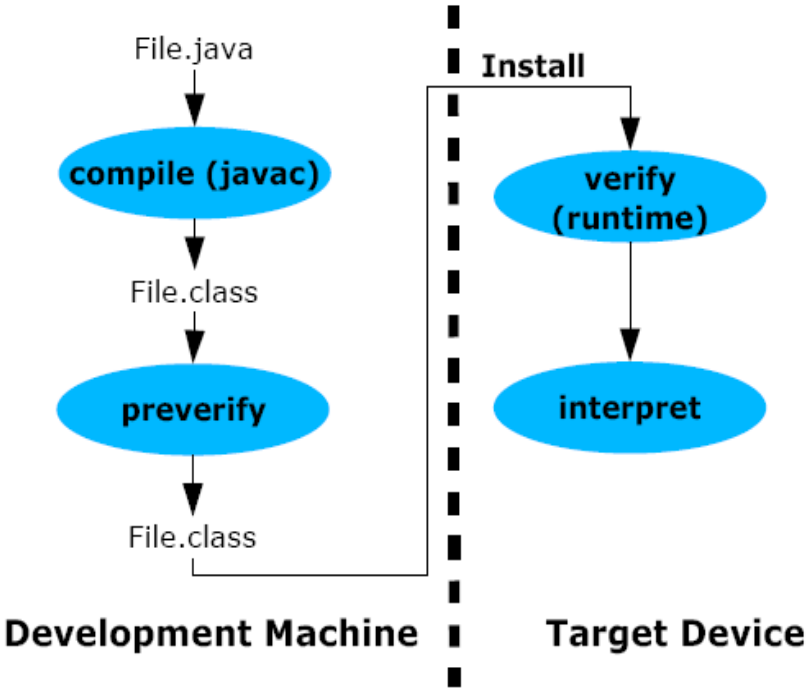
Nói chung CLDC không định nghĩa các ứng dụng cài đặt, giao diện người dùng (UI) và các sự kiện handling. Còn MIDP định nghĩa chu trình sống của ứng dụng MIDP (MIDlet), thư viện UI, và các sự kiện handling nằm trong (javax.microedition.lcdui.*)

Class verification – Kiểm tra các lớp

CLDC yêu cầu tất cả các lớp phải trải qua hai pha kiểm tra

- Pha kiểm tra đầu tiên được thực hiện off-device trước để cài đặt trên thiết bị.
- Pha thứ hai xuất hiện on-device trong suốt thời gian chạy và được thực hiện bởi KVM

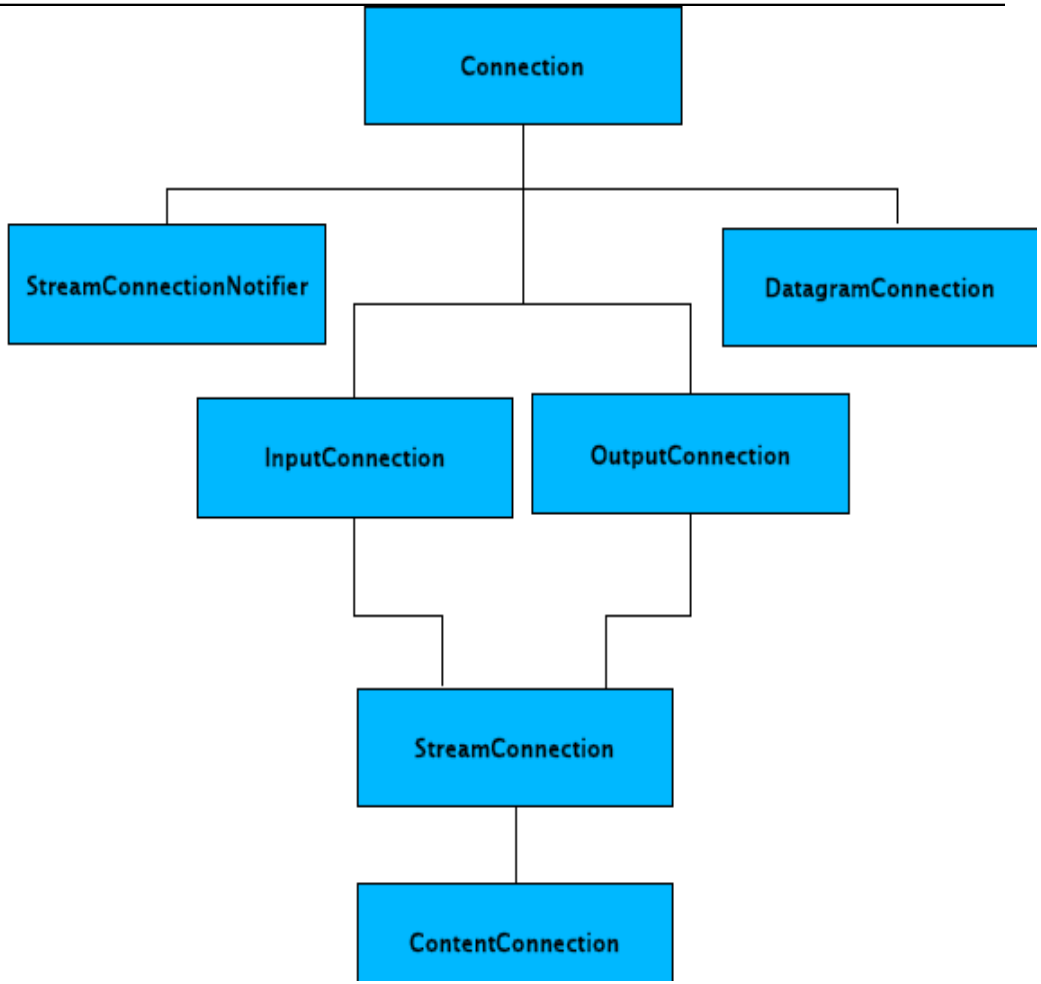
2 pha kiểm tra



Generic Connection Framework
- khung kết nối chung - GCF

GCF – cung cấp API cơ bản cho các kết nối trong CLDC. Khung này cung cấp cơ sở chung cho các kết nối như HTTP,Sockets và Datagram.

GCF cung cấp một tập các API trừu tượng cho tất cả các kết nối này



CDC

- Connected Device Configurarion – CDC – Cấu hình kết nối thiết bị - là một tập cha của CLDC. Nó cung cấp một môi trường runtime rộng hơn CLDC và nó gần với môi trường J2SE
- Máy ảo Java CDC (CVM) là một máy ảo đầy đủ. CDC chứa tất cả API từ CLDC. Nó cung cấp một tập lớn hơn các lớp của J2SE
- Tương tự CLDC, CDC không định nghĩa bất kỳ lớp giao diện người dùng (UI) nào. Các thư viện UI được định nghĩa bởi profile
- CDC cũng chứa GCF

JWTI - Java Technology for the Wireless Industry (JWTI)

JWTI – công nghệ JAVA cho mạng không dây- xác định một tập hệ thống dịch vụ và các mô tả chuẩn phục vụ cho cực tiểu hóa các ứng dụng phân mảnh trong các thiết bị mobile

MIDP

- Mobile Information Device Profile (MIDP) – được xây dựng bên trên đỉnh của CLDC. Không có ứng dụng mobile nào có thể viết bằng cách chỉ sử dụng API CLDC. Do đó trong MIDP những ứng dụng giao diện người dùng API được định nghĩa.
- Các mô tả MIDP, tương tự CLDC và các API khác được định nghĩa qua quá trình giao tiếp JAVA.
- Các mô tả MIDP định nghĩa một thiết bị MID có các đặc tính tối thiểu sau:

Màn hình hiển thị:

- Screen-size (kích cỡ màn hình) : 96x54
- Display depth (Độ sâu hiển thị) : 1-bit
- Tỉ lệ Pixel : approximately 1:1
- Input: user input, one-handedkeyboard, hay touch screen
- Bộ nhớ: 256 Kb cho MIDP
  8Kb cho dữ liệu
  128 Kb cho môi trường chạy JAVA

Mạng:

- Wireless, băng thông hạn chế
- Âm thanh: Có khả năng chơi nhạc…

Tóm lại : MIDP định nghĩa mô hình ứng dụng, giao diện người dùng, lưu trữ cố định, mạng, trò chơi đa phương tiện…

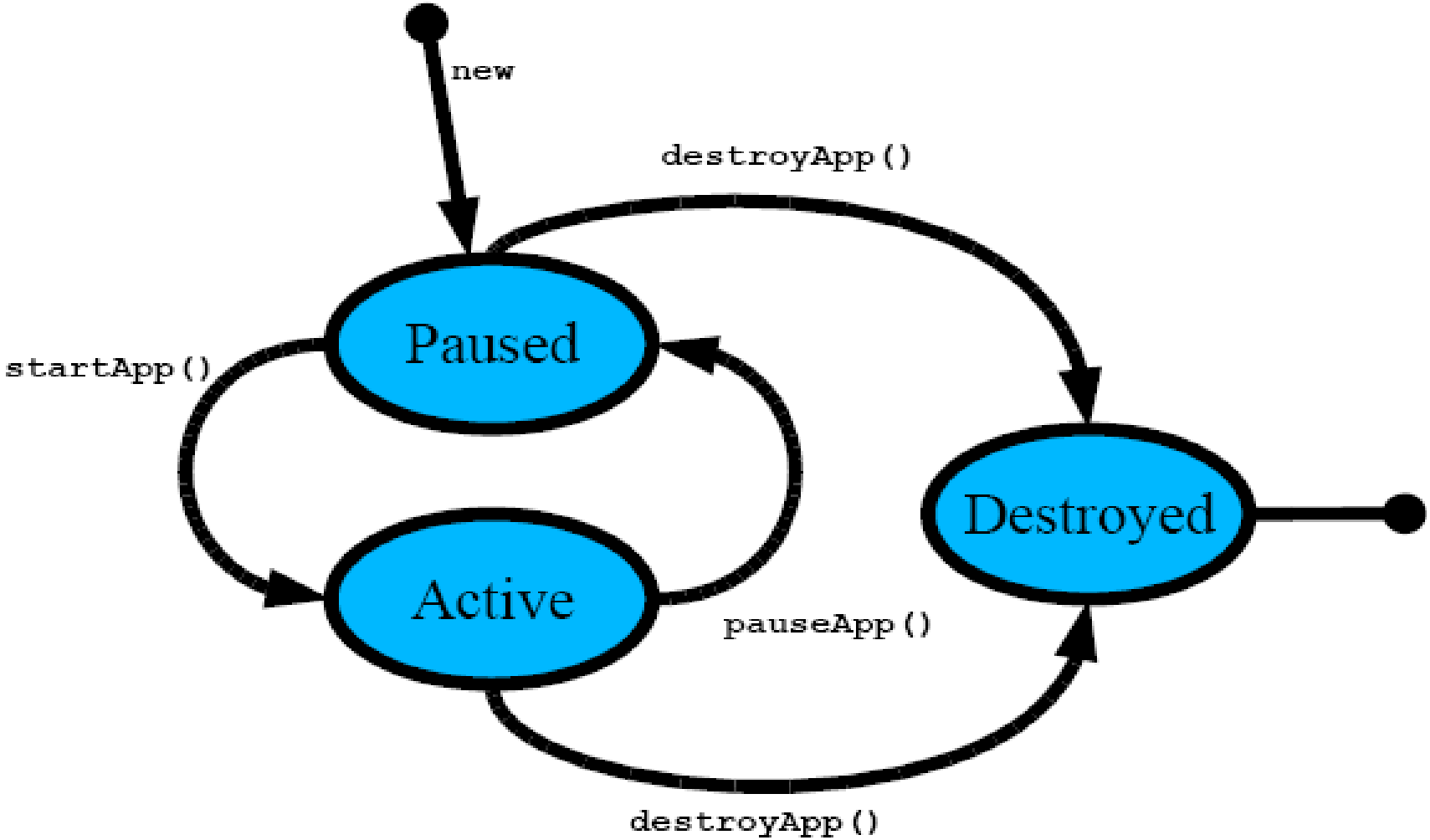
MIDlet

- Một ứng dụng MIDP được gọi là một MIDlet. Phần mềm quản lý ứng dụng của thiết bị (AMS) tương tác trực tiếp với MIDlet thông qua các phương thức tạo (create), bắt đầu (start), tạm dừng (pause) và hủy bỏ (destroy) của MIDlet
- MIDlet là một phần của gói javax.microedition.midlet.
- Một MIDlet phải mở rộng từ class MIDlet. Nó có thể yêu cầu các tham số từ AMS được định nghĩa trong mô tả ứng dụng (JAD)
- Một MIDlet không có phương thức public static void main (String[] argv)

Chu trình sống của MIDlet

- Chu trình sống của MIDlet bắt đầu khi nó được khởi tạo bởi AMS. Sau khi được tạo ra, MIDlet rơi vào trạng thái Paused - tạm dừng để chờ lệnh mới. Sau đó AMS gọi hàm khởi tạo constructor không tham số của MIDlet. Nếu có một loại trừ xuất hiện tại constructor, MIDlet được đưa vào trạng thái Destroyed và được loại bỏ ngay lập tức
- MIDlet chuyển vào trạng thái Active trong suốt thời gian AMS gọi phương thức Startup()
- MIDlet rơi vào trạng thái Destroy khi AMS gọi phương thức destroyApp(). Chú ý MIDlet rơi vào trạng thái Destroy chỉ một lần trong suốt chu trình sống của nó

Chu trình sống của MIDlet



Đóng gói MIDlet

- Ứng dụng MIDP được đóng gói và chuyển vào thiết bị như một gói MIDlet. Một gói MIDlet bao gồm một file JAR (JAVA Archive) và một file tham số JAVA Application Descriptor (JAD)
- Một file JAD là một file text chứa một tập các thuộc tính và một vài yêu cầu khác

Lập trình với Mobile

Sau bài này SV sẽ nắm được các vấn đề sau

- Tạo một MIDlet đơn giản
- Tạo một Project trong Netbean
- Tạo một MIDlet trong Netbean
- Chạy một MIDlet trong điện thoại mô phỏng

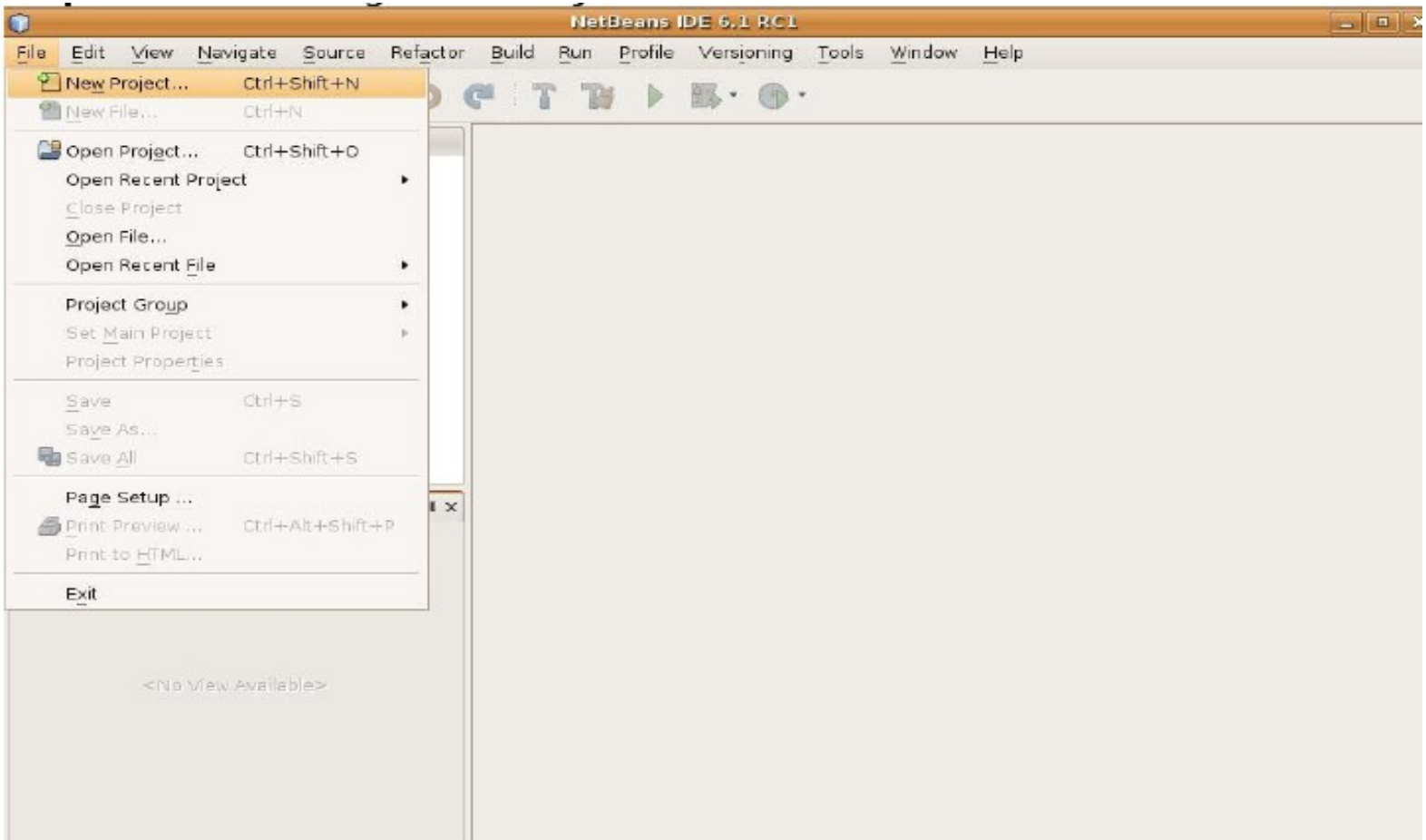
Giới thiệu

Một IDE – Integrated Devepment Enviroment – là một môi trường lập trình với một môi trường xây dựng GUI, một màn hình soạn thảo, một bộ biên dịch và gỡ rối. Trong phần nghiên cứu này chọn Netbean đáp ứng với nhu cầu trên thiết bị mô phỏng, điều này giúp chương trình mô phỏng của chúng ta trông giống với thiết bị thực

Chuẩn bị : Bộ cài netbean 6.1 RC và Mobility Pack

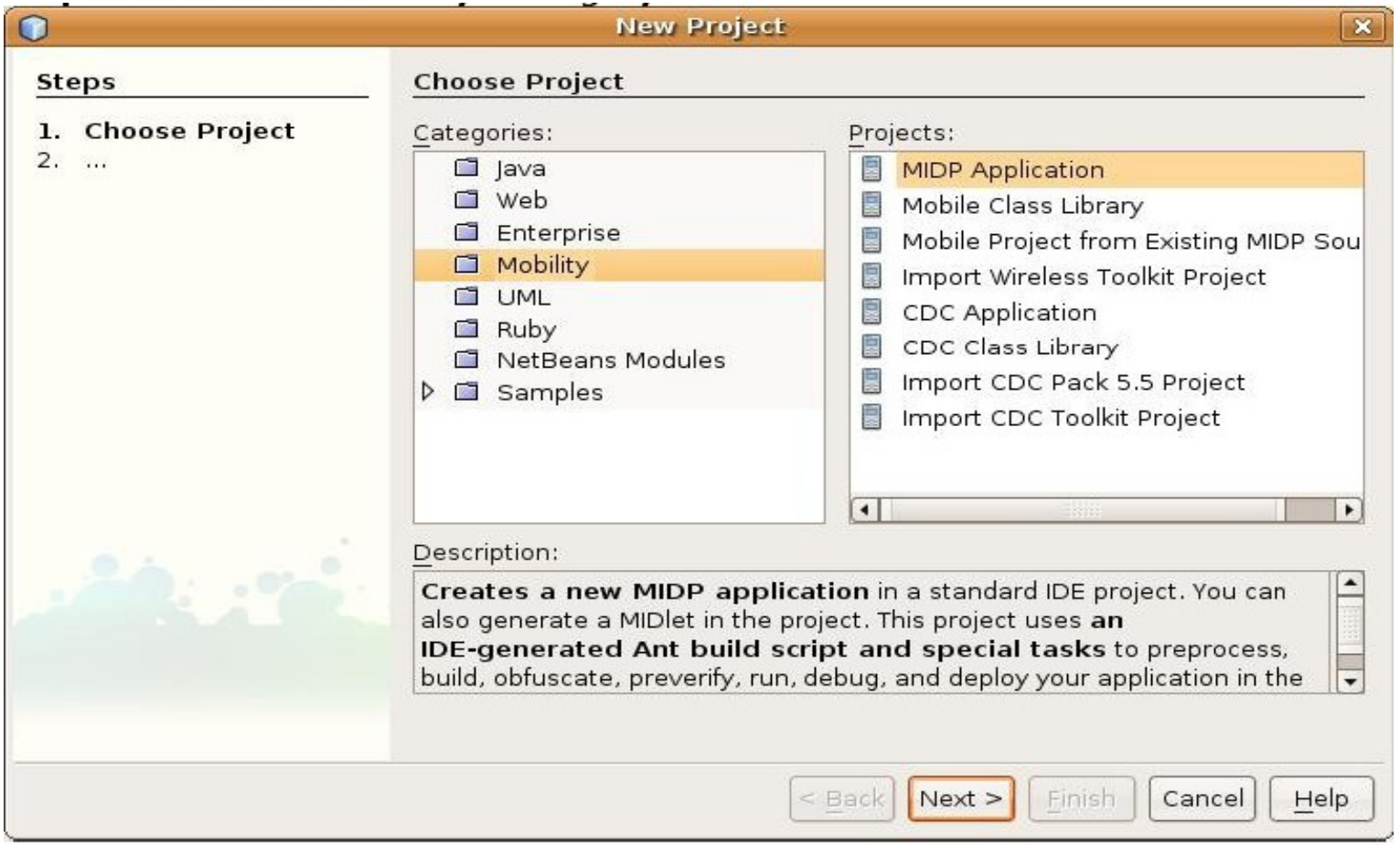
Sử dụng Netbean và Mobility Packet

Step1 Bắt đầu khởi tạo một project

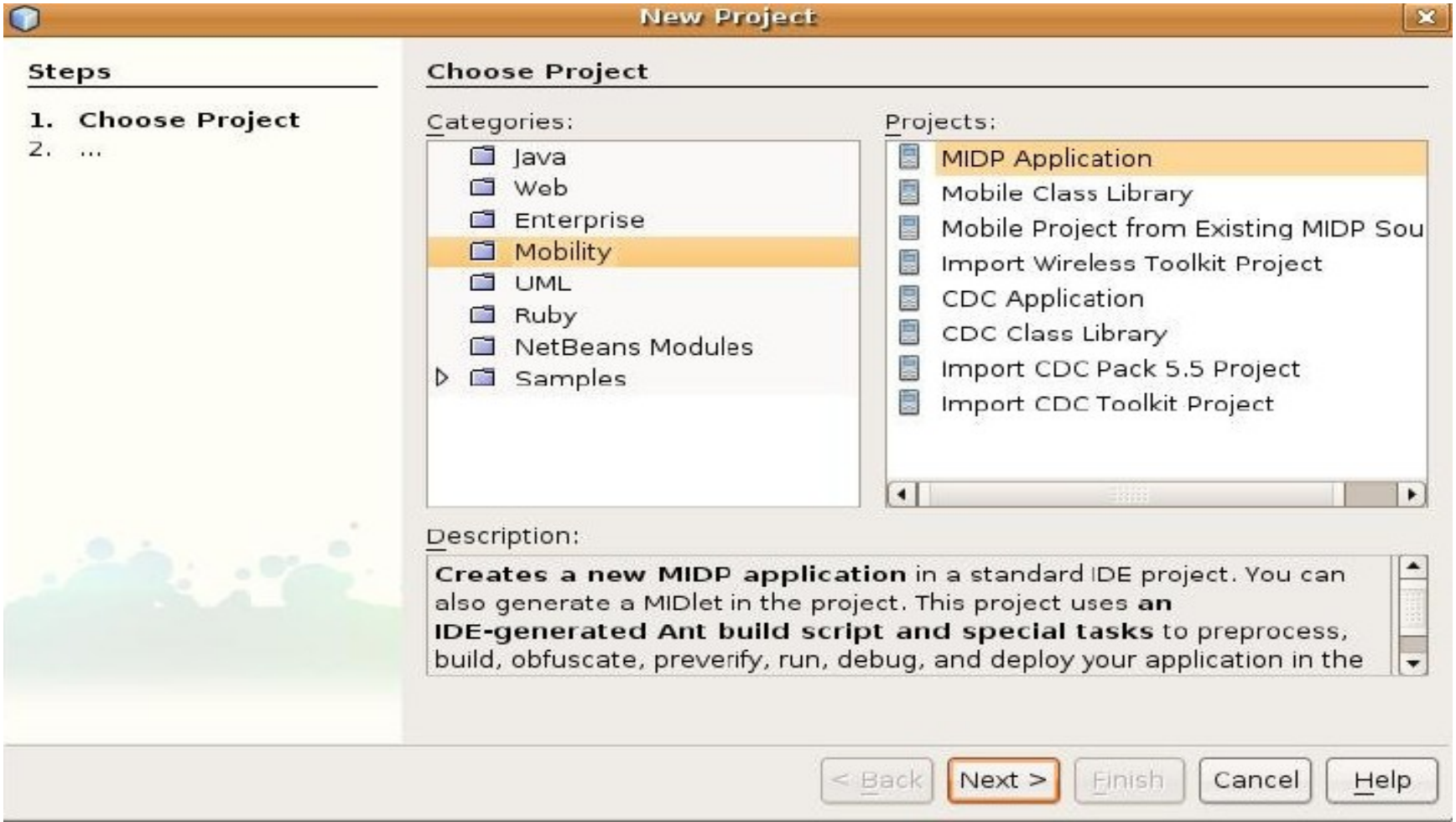


Hello Word MIDlet

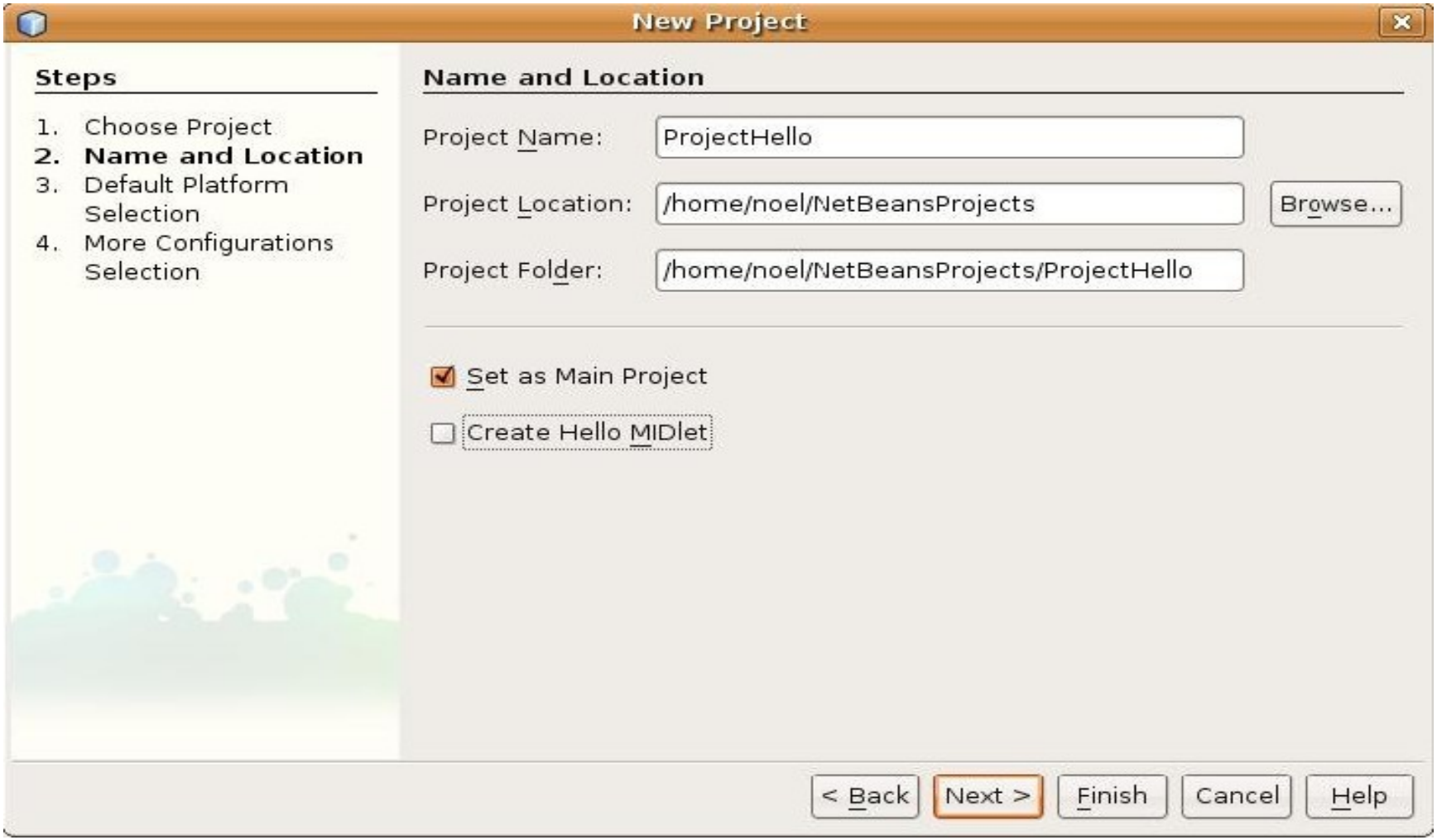
Step 2 Lựa chọn loại Mobile



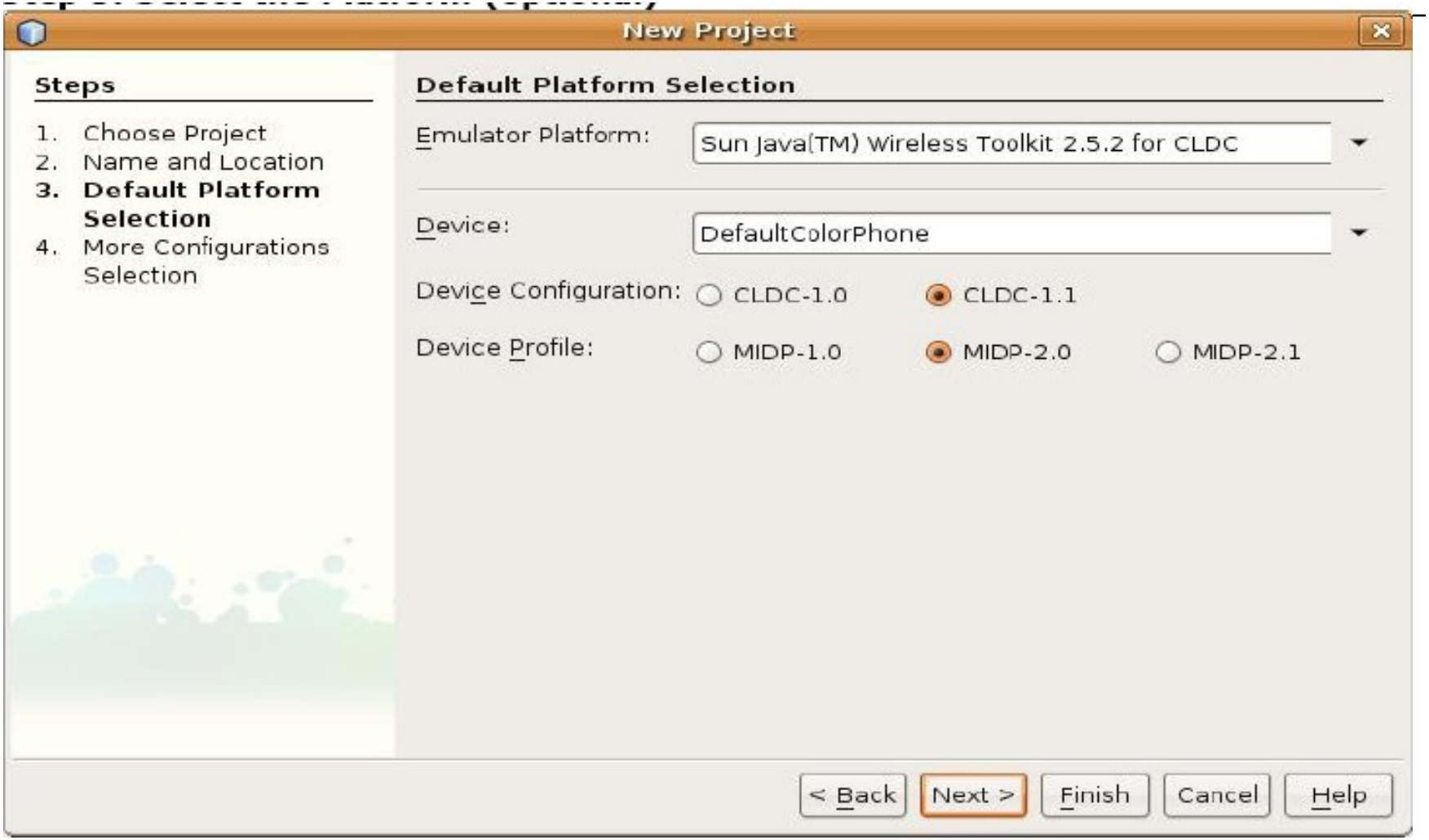
Step 3 Lựa chọn ứng dụng MIDP



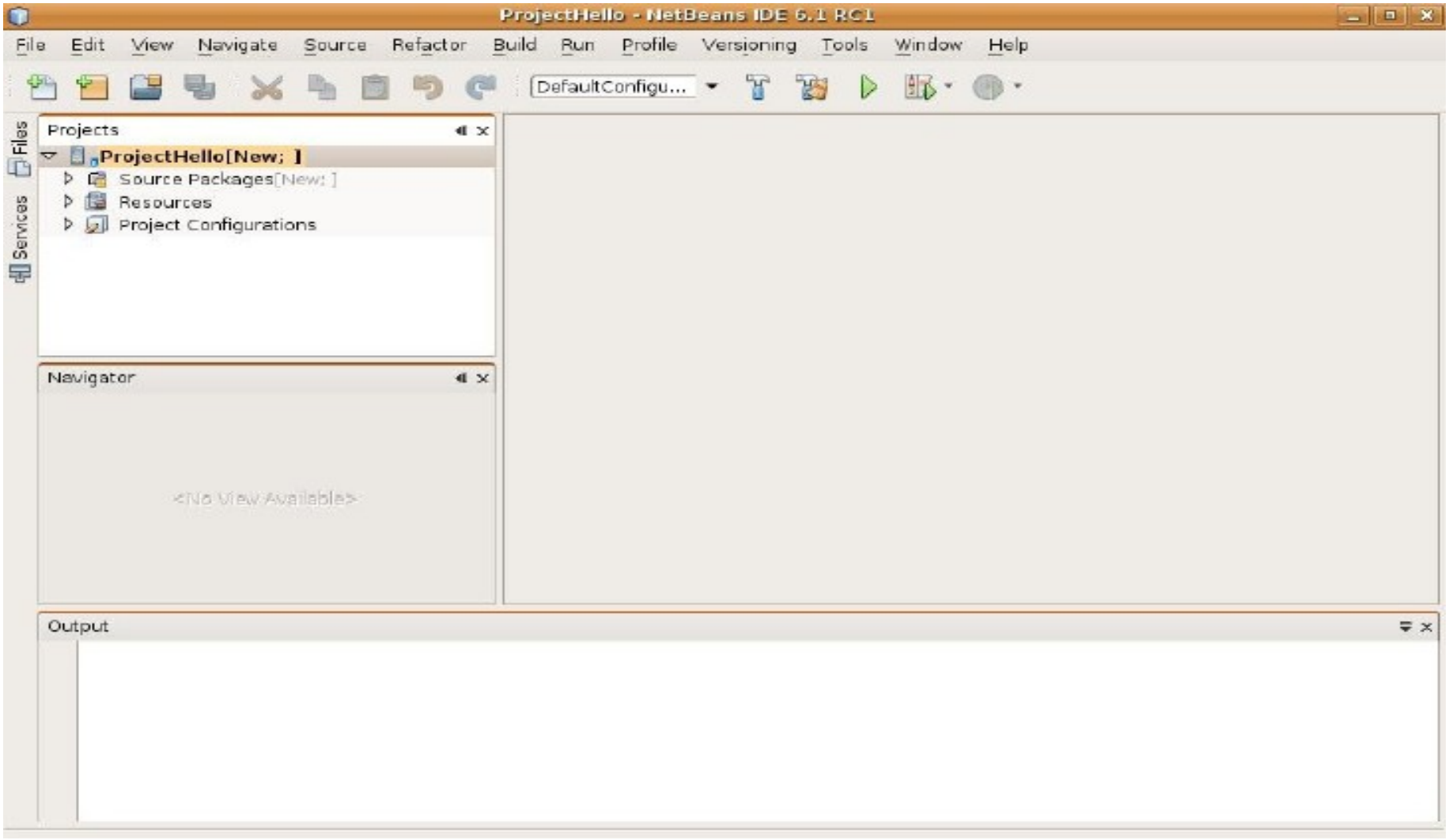
Step 4 Đặt tên Project và xác định vị trí lưu trữ



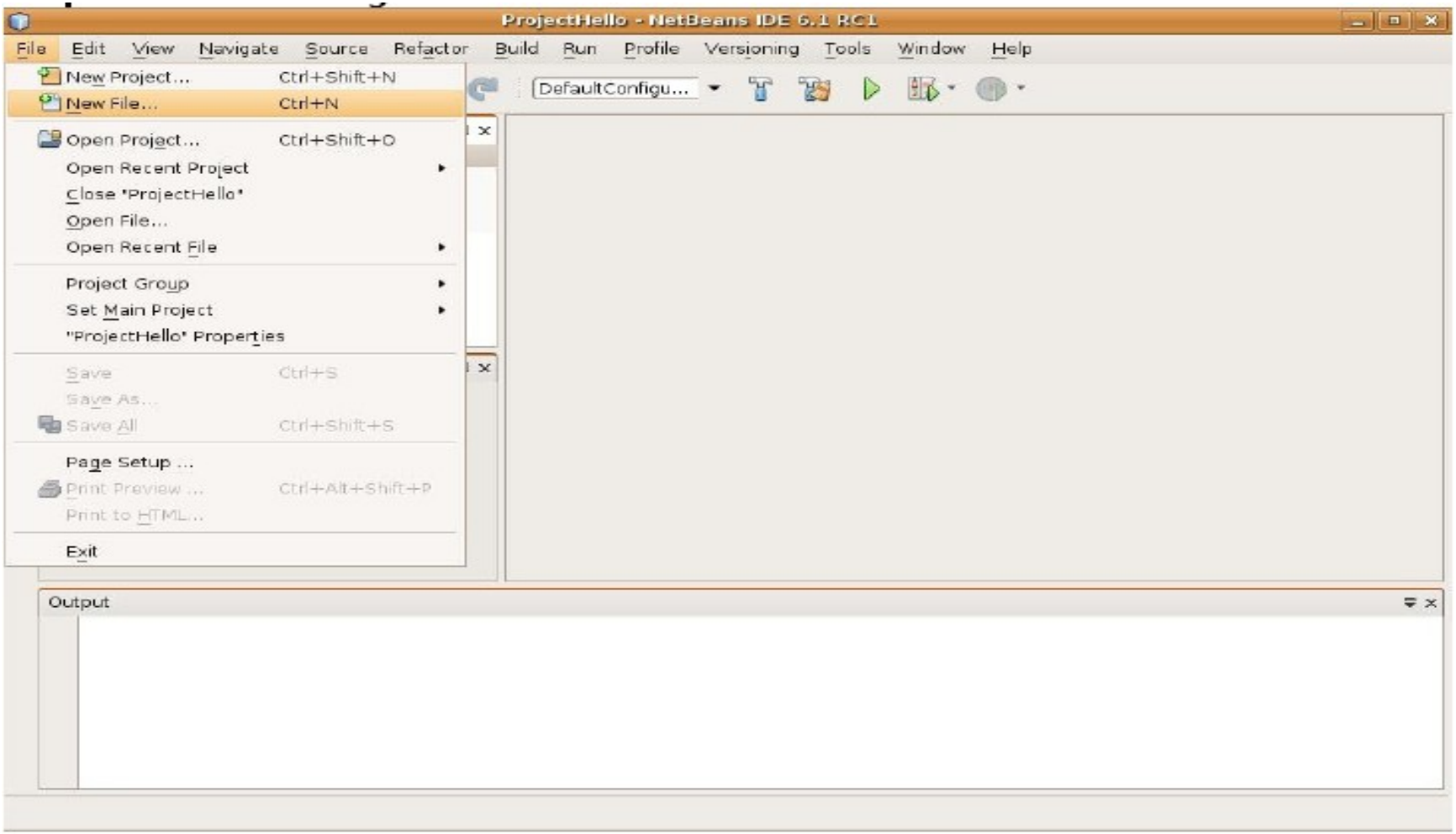
Step 5 Lựa chọn Platform (tham số)



Một project mới được tạo ra trong Netbean

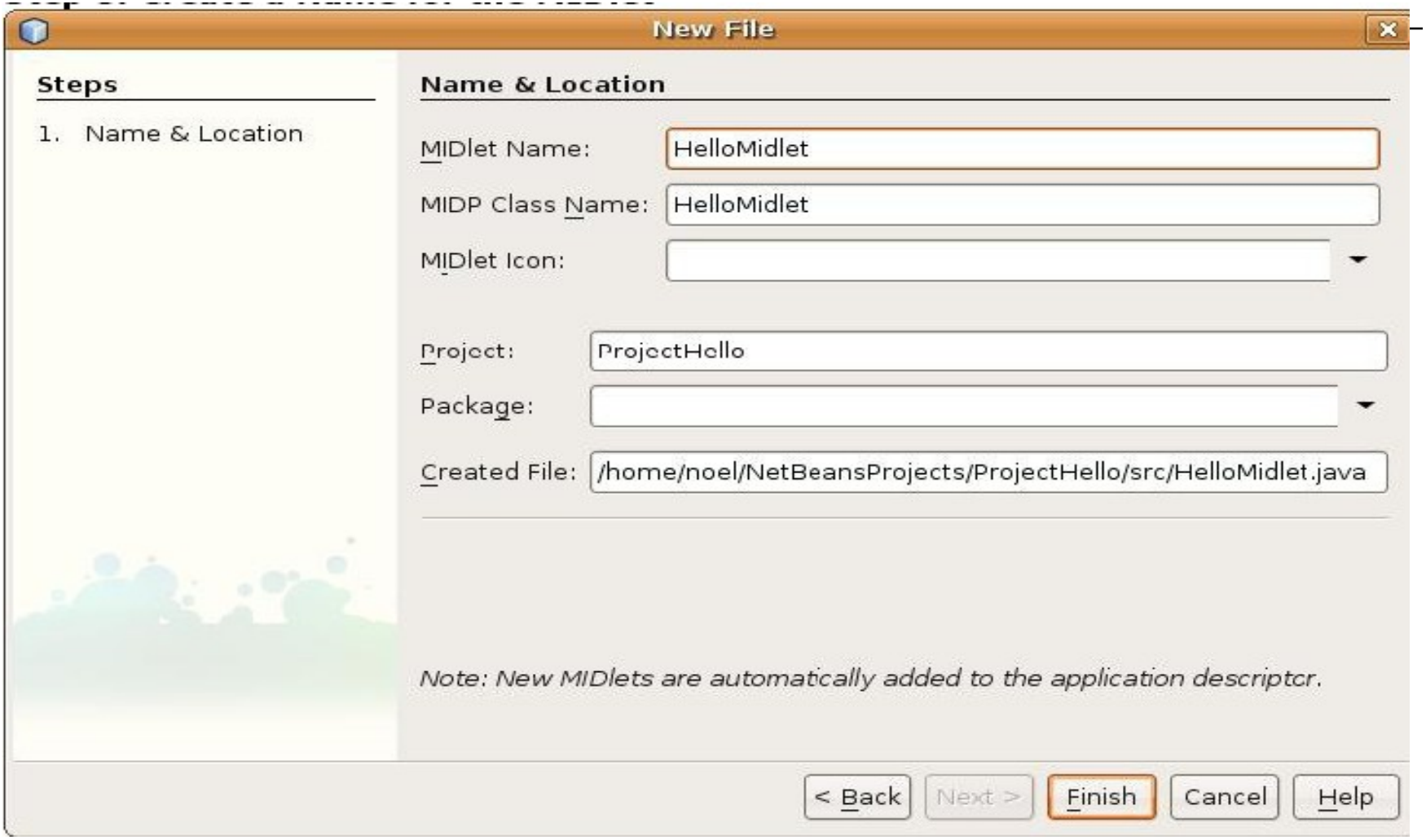


Step 6 Tạo một MIDlet

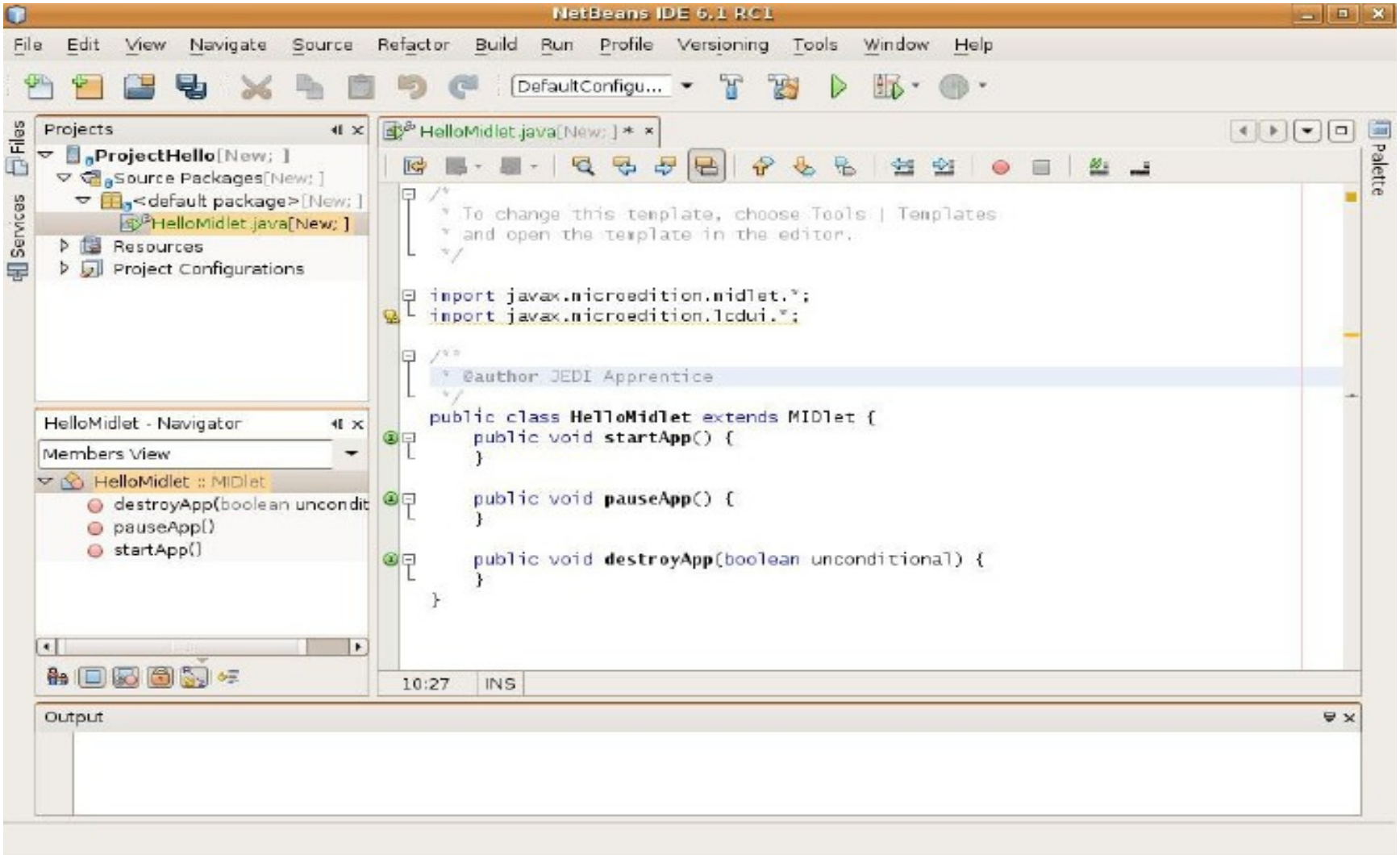


Step 7 Lựa chọn loại MIDP và kiểu MIDlet

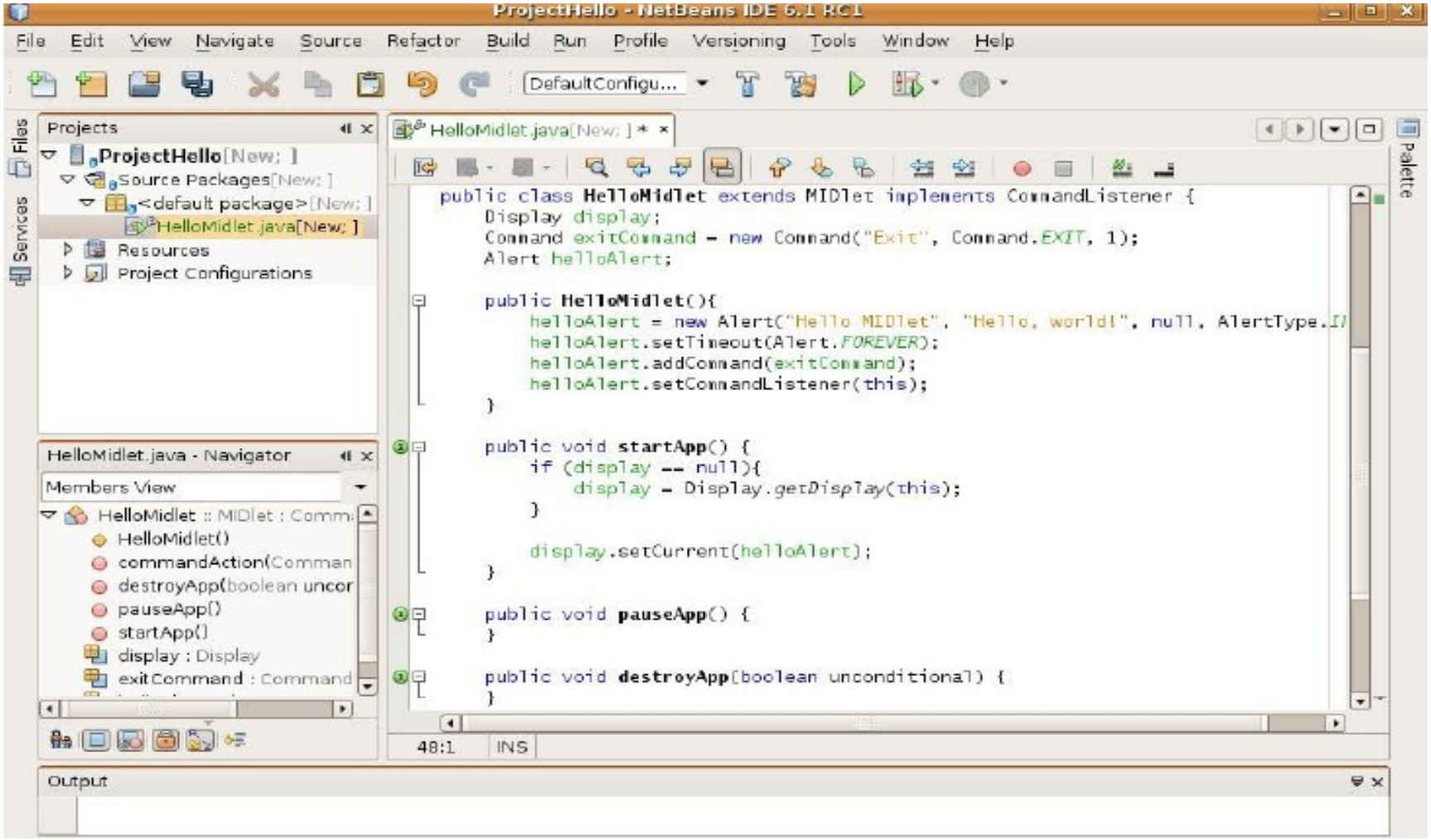
Step 8 Đặt tên cho MIDlet



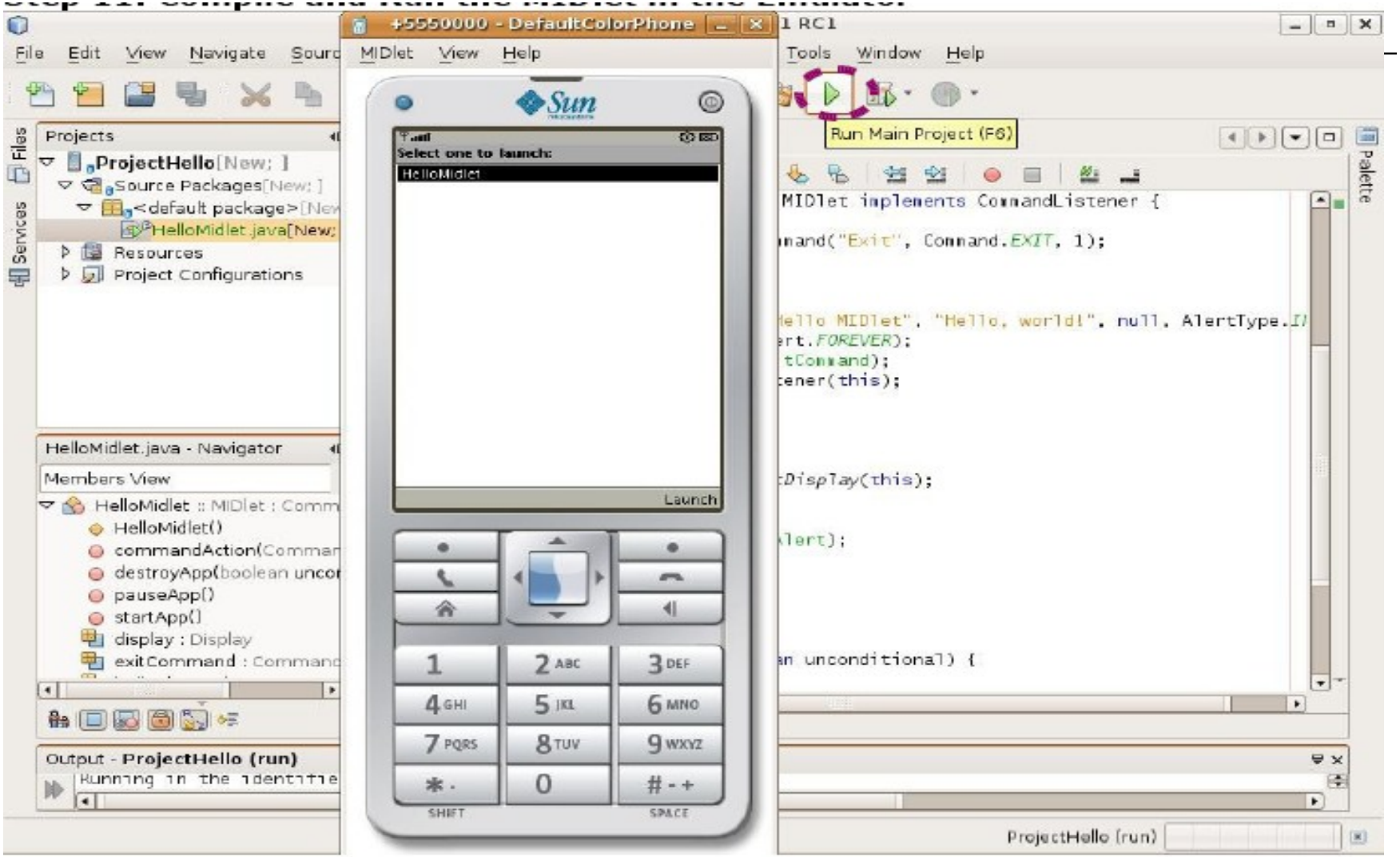
Một MIDlet mới tự động tạo ra các phương thức yêu cầu



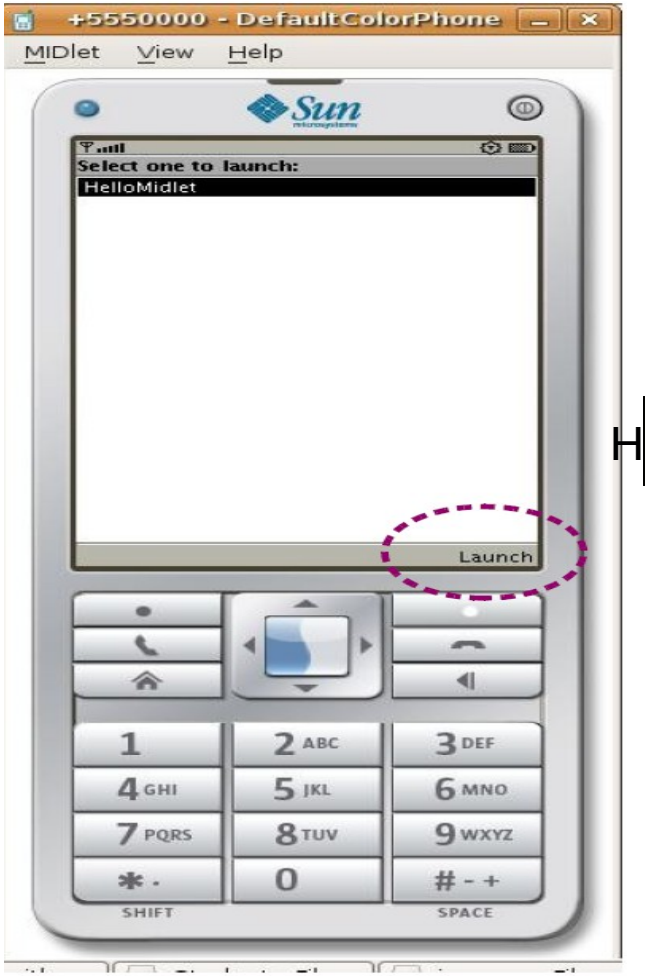
Step 10 Thay thế các code tự động được tạo ra bằng code của chúng ta



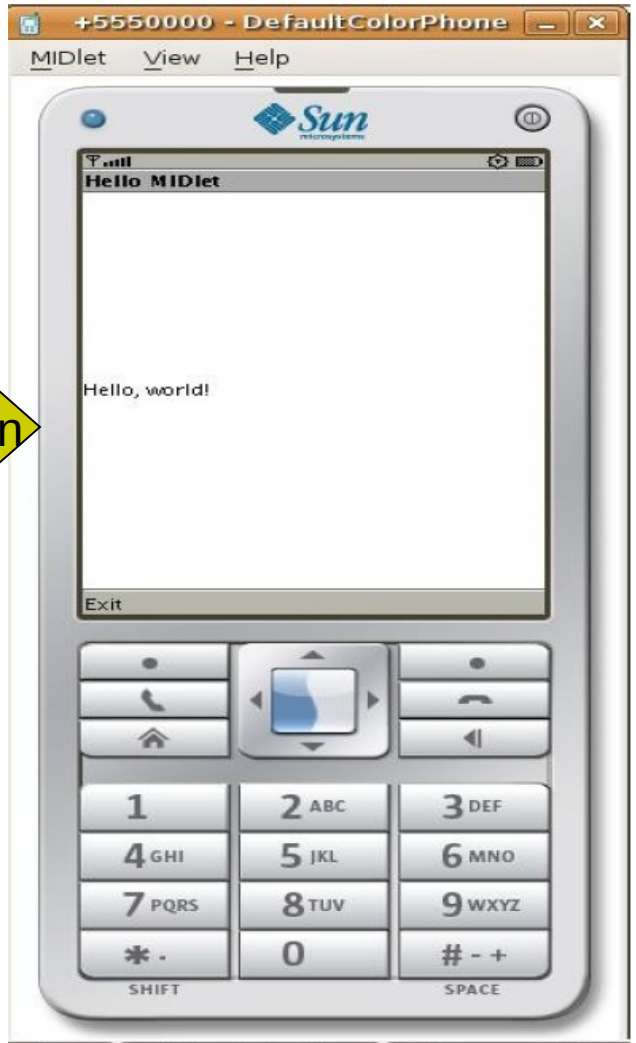
Step 11 Compile và Run MIDlet trong mô phỏng



Step 12 Chạy MIDlet của chúng ta trong mô phỏng



Hello word in action



Hello word MIDLet

- Chu trình sống của một MIDlet bắt đầu khi nó được tạo ra bằng một hệ thống quản lý ứng dụng AMS của thiết bị. Khởi tạo, nó trong trạng thái Paused.
- Để có thể tạo một MIDlet chúng ta phải tạo một lớp con subclass của lớp MIDlet từ gói javax.microedition.midlet.