Unit 7

Abstractions for programming

Discussion Topics

- Abstractions Levels: Libraries, System Software, Toolkits, High level Programming Languages

Prepared by: Shiv Raj Pant

What is abstraction?

- In computer science, abstraction is a technique for arranging complexity of computer systems.
- It works by establishing a level of complexity on which a person interacts with the system, suppressing the more complex details below the current level.
- It's often described as the creation of well-defined interfaces to hide the inner workings of computer programs from users.
- It's the process of identifying the general characteristics needed to solve a problem while filtering out unnecessary information. Abstractions "simplify a process or artifact, by providing essential things, and hiding the (possibly) useless details.“

  e.g. OS is an excellent example of abstraction. A user is only concerned with working with application on high level and doesn’t care about the hardware details. For example, while working on business data on spreadsheet, we just create a new file, store our data and save the file. The opening/saving is really a complex process on hardware level. But OS does all that for us.

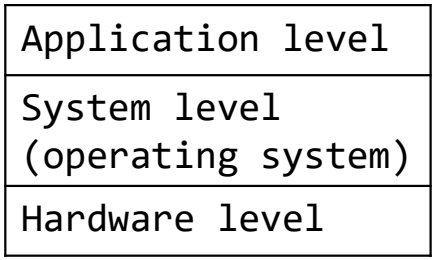


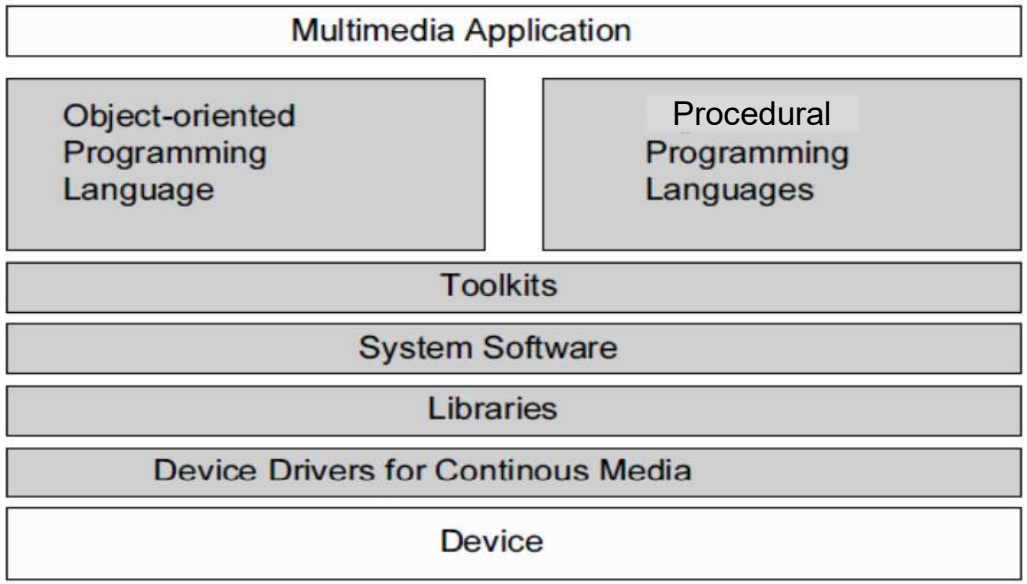
Fig: General levels of abstraction in computing systems.

Abstractions for programming multimedia systems?

- Multimedia applications can be implemented in procedural as well as OO languages
- The application code for multimedia application programs may be hardware-dependent. However, this problem can be mitigated by using common OS extensions (another abstraction!).
- There are different programming possibilities for accessing and representing multimedia data.

Abstraction levels

- Abstraction levels in programming define different approaches with a varying degree of detail for representing, accessing and manipulating data.
- The abstraction levels w.r.to multimedia data is shown in figure.



Abstraction Levels of the Programming of Multimedia Systems

1. Device (Hardware):

- This is the lowest level of abstraction(or no abstraction at all).
- It deals with the real physical devices for processing media.
- Hardware is separate physical component which is accessible to applications.
- Here, by "hardware" we mean the interface devices connected to I/O devices. e.g., sound card or video card.

2. Device drivers:

- It is the low level abstraction.
- It is the part of the system software that facilitates communication between operating system and actual device.
- Since different devices are made differently, the operating system can not recognize the hardware-level details of the devices.
- The OS recognizes particular hardware structure through device driver.
- In other words, the multimedia devices are bound to OS through device driver.

3. Libraries:

- Libraries are higher level abstraction for programmers.
- Libraries involve a set of functions for processing of continuous media.
- The libraries are provided together with corresponding hardware.
- Libraries hide the intricate details of the hardware and provide a higher-level abstraction for programmer that is easy to use.
- Problem: Libraries are hardware dependent.

4. System software:

- Instead of implementing access to multimedia devices through individual libraries, the device access can become part of the operating system.
- This level of abstraction provides programmers implement access to multimedia devices and media processing through OS-level services.

- At the system level, the multimedia data is represented as "stream".
- E.g. in MS Windows, a Media control interface (MCI) provides the interface for processing multimedia data. It allows access to media streams and their corresponding devices.

5. Toolkits:

- Toolkits are similar to libraries but with a higher level of abstraction.
- Toolkits are softwares used to abstract from actual physical device. This is also done by libraries in very limited way.
- Toolkits allow uniform interface for communication with all different devices of continuous media.(whereas libraries are hardware-dependent)
- Toolkits can also hide the process-structures.
- They represent interfaces at the system software level. So they can be embedded in high-level programming languages.

6. High-level Programming languages:

- HLPL are the highest-level abstractions from the programmer's point of view.
- High level languages (procedural and object-oriented) provide hardware-independent functions (or class/methods) to support and manipulate multimedia data.
- The programs in HLL either directly access multimedia data structures or communicate directly with active OS processes in a real-time environment. The processing devices are controlled through corresponding device drivers and Compiler provides the communication between application program and the processing of continuous data.

More detail on the abstraction levels

1. Libraries:

- Libraries are abstraction where the processing of media is based on a set of functions. The set of primitive functions are embedded into libraries.
- The libraries are provided together with corresponding hardware.
- Libraries hide the intricate details of the hardware and provide a higher-level abstraction for programmer that is easy to use.
- The libraries may differ in their degree of abstraction:
  - Some libraries are extensions of GUI
  - Some libraries consist of control instructions passed as control blocks to the corresponding driver.
- There are different libraries for different interfaces. i.e., Libraries are hardware dependent.
- Libraries are very useful at the operating system level.
- However, there is no standard about which functions are best for different drivers i.e. which functions should be supported.
- Therefore, there are variety of interfaces and hence, a set of different libraries.
- Libraries can be made hardware-independent by either support of OS for media or by integration in the high level programming environment.

2. System software:

- Instead of implementing access to multimedia devices through individual libraries, the device access can become part of the operating system.
- This level of abstraction provides programmers implement access to multimedia devices and media processing through OS-level services.
- An example of access of multimedia devices and support for continuous media processing implemented in OS is Nemo system.
- There are two ways to represent data in this approach:

Data as time capsules

- Time capsule is special abstractions related to files system.
- The special file extensions serve as modification and access for continuous media.
- The data is represented as collection of LDUs(logical data units).

- Each LDU has a time capsule which keeps information about data type, actual value of data, and its life span.
- E.g. if a video has 25 frames per second, each frame has a valid life span of 40 ms.

Data as streams

- A well known, used and implemented abstraction at the system level is the stream.
- A stream denotes the continuous flow of audio and video data.
- Initially, a stream is established between source and sinks. Then, operations on a stream can be performed such as *play, fast forward, rewind and stop*.
- E.g. in MS Windows, a Media control interface (MCI) provides the interface for processing multimedia data. It allows access to media streams and their corresponding devices.

3. Toolkits:

- Toolkits are similar to libraries but with a higher level of abstraction.
- It is a simpler approach (from the user perspective) in a programming environment than the system software interface for control of the audio and video data processing.
- The toolkits are used to:
  - Abstract from actual physical device. (This is also done by libraries in very limited way.)
  - allow uniform interface for communication with all different devices of continuous media.(whereas libraries are hardware-dependent)
- Toolkits can also hide the process-structures.
- They represent interfaces at the system software level. So they can be embedded in high-level programming languages.

4. Procedural programming languages

- HLPL are the highest-level abstractions from the programmer's point of view.
- High level procedural languages provide hardware-independent functions to support and manipulate multimedia data.
- Since, the function calls in HLPL are mostly hardware- and driver-independent, their integration in HLPL gives a good abstraction.
- For this approach to work, the programs must be capable of supporting and effectively manipulating multimedia data.
- Here, we do not necessarily need a different dedicated language for multimedia. Any existing general-purpose language can be extended (by using libraries) to support multimedia.

Representation of media in a HLPL

- The programs in HLL either directly access multimedia data structures or communicate directly with active OS processes in a real-time environment. The processing devices are controlled through corresponding device drivers and Compiler provides the communication between application program and the processing of continuous data.
- Therefore, media can be represented in three different ways inside a programming language:

i. Media as "data types"

ii. Media as "files"

iii. Media as processes

i. Media as "data types"

- In this approach, we can use special "data types" for multimedia data (just as other data types - int, struct etc)
- For text data, character is the data type. Similarly for audio/video, the smallest unit can be defined as LDU (logical data unit).
- Following is an example of using media as data type. Here two LDUs from microphones are read and mixed.

```
AUDIO_LDU ldu.left1,ldu.left2,ldu.left_mixed
input(micro1,ldu.left1)
input(micro2,ldu.left2)
ldu.left_mixed = ldu.left1+ldu.left2
```

ii. Media as "files"

- In this approach, we consider media streams as files instead of data types.
- In his approach, a device unit, which creates or processes media is associated with a file name.
- Several functions associated with the media file can be implemented such as read, write, seek, etc.

Example:

```
file_h1 = open(MICROPHONE_1,.. ..)
file_h2 = open(MICROPHONE_2,.. ..)
file_h3 = open(SPEAKER, .. ..)
.. ..
read(file_h1)
read(file_h2)
mix(file_3,file_h1,fileh2)
close(file_h1)
... ..
```

ii. Media as "processes"

- In this representation, media is represented as processes and integrated into HLPL
- Each process implements as set of actions(functions) which apply to a continuous data stream.
- During the creation of process, the used physical device is identified and locked. The actions of the process are controlled through IPC mechanism.

```
PROCESS cont_process_a
...
On_message_do
   Set_volume .. ..
   Set_loudness .. ..

main()
pid = create(cont_process_a)
send(pid, set_volume,3)
send(pid, set_loudness)
.. ..
```

5. Object-oriented programming language approach

- OO approach was introduced for reducing the complexity of the real world applications development.
- OO approach remains a natural way of representing data as real-world objects.
- The main features of OO approach are classes/objects, reuse, inheritance, polymorphism, data encapsulation etc..
- OO languages provide programmers a better abstraction due to its natural representation of real-world.
- The programs are implemented by using classes, objects and method instead of using functions and data structures.

Object oriented concepts for multimedia

Abstract data type

Class

Object

Inheritance

Polymorphism

End of unit 7

Thank you!