Inferno Operating System

Lecture on Inferno Operating System

Olugbenga O. Fadiya

Dec 14, 2011

Loyola University Chicago, Computer Science

Introduction

Short Lecture on Inferno Operating System

by

Olugbenga O. Fadiya

Outline

- Brief History
- Principles of Inferno OS
- Architecture
- How it is being used
- Conclusion
- Questions

What is Inferno?

- Distributed Operating System developed and first released in 1996 by the Computing Science Research Center of Bell Labs for creating and supporting distributed services.
  - Derived from Plan 9 and originally developed as a commercial product but now an open source software managed by VitaNuova with the latest release in 2007.
  - Shares basic principles with Plan 9: Resources as files, Namespace and Standard Communication protocols.
  - Runs as a native OS or as an application which provides virtual operating over other platforms.
  - lightweight, portable and versatile OS which cuts across several dimensions.

The Inferno OS

- Written in C programming language to support applications written in Limbo Langauge.
- Operating system built on distributed computing.
- All resources represented as file.
- Dynamically adjusted namespace system.
- Single communcation protocol for all resources.

Principles of Inferno OS

- Resources as Files:
  - Views all resources (external and internal) as files.
  - Access and manipulation of windows file by puck*ls*/*dev*/*draw*/2.
  - Access to TCP/IP through puck*ls*/*net*/*tcp*/0.
  - All Inferno resources, both local and remote such as storage devices, processes, services are all represented as a file accessible by any application by manipulating the relevant files required using standard operations.

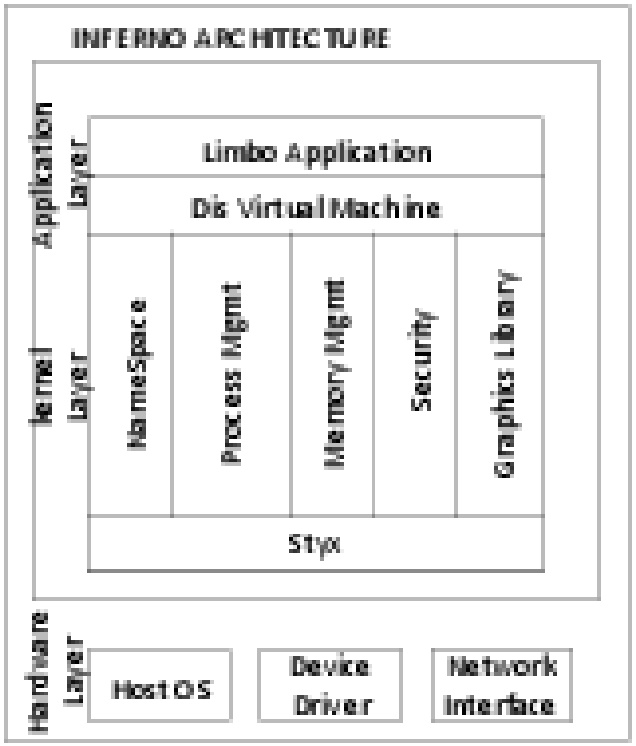
Principles of Inferno OS

- Namespaces:
  - Presents the application view of the network as a single, coherent namespace.
  - Namespace appear as a hierarchical file system but represents physically separated (locally or remotely) resources.
  - Application able to use services and/or resources transparently by building a unique private view of the resources and services it needs to access with each set of resources represented as a hierarchy of files and accessible via the standard file access.

Principles of Inferno OS

- Standard Communication Protocol:
  - This protocol is a file service protocol called 9P (an earlier variant was called Styx).
  - The representation of all resources as files makes it possible to utilized only one protocol to communicate with and provide resources,
  - Namespace appear as a hierarchical file system but represents physically separated (locally or remotely) resources.

Inferno Architecture



Kernel Layer

- The kernel layer is responsible for the connection of the application layer to that of the hardware. Memory and Process management occurs at this layer within Inferno OS, with programs in Inferno using the time-share feature it possesses; each program has independent access to memory.
- Memory Management:
  - Unique concept of memory allocation.
  - No fixed assignment of components in memory space.
  - Memory allocated from the main, heap and image pool.
  - Memory allocation made using the malloc ( ) call but all calls handled by dopoolalloc ( ) defined in the emu /port/alloc.c.
  - Example of this function (code borrowed)
  - static void *dopoolalloc (Pool *p, ulong asize, ulong pc)
  - p here points the Pool structure, asize gives the size of the allocation requests in bytes and pc is the program counter of the caller.

Kernel Layer

- Process management also occurs at this layer in Inferno though unusual in comparison to other traditional OS owing to Infernos use of a virtual machine, this VM interpreter which is embedded in the OS has more control over user programs than that of a typical OS.Kernel process state in hosted Inferno are handled by the host OS in contrast to native Inferno where it is responsible for managing these process.
- Process Management:
  - All process identified as threads.
  - Share common memory space.
  - Single multithreaded program concept.
  - kproc ( ) function use is peculiar.

Hardware layer

- Provides device driver interface which programs use to connect to the underlying hardware.
- This process handled by creating or implementing a file tree with a file name, this name is bound to locations in the namespace which provides the files of the console device.

Application layer

- All applications written in Limbo Language dynamically.
- Limbo applications, built of modules and are/can be assessed dynamically.
- Limbo applications more resticted than those built in C language, offer boundary checks which prevent the vulnerability against buffer flow attacks.

How it is being used

- Network gateways
- Advanced hand-held devices
- TV set-up boxes
- Cable television
- Direct satellite broadcast

Conclusion

- Program built on a distributed model
- Everything is a file
- Limbo application supports multitasking and scheduling threads of control.
- Communication protocol independent of any underlying network.
- Minimal hardware requirement, can run on as little as 1MB of memory.
- Poor Latency.

Conclusion

Supplemental Reading

Inferno Operating System

by

Olugbenga O. Fadiya

Conclusion

THANK-YOU

Conclusion

Questions