Portable Inter process Communication Programming

Morteza Kashyian
Computer Engineering Department
Iran University of Science & Technology
kashyian@gmail.com

Seyedeh Leili Mirtaheri
Department of Computer Engineering
Iran University of Science & Technology
mirtaheri@comp.iust.ac.ir

Ehsan Mousavi Khaneghah
Department of Computer Engineering
Iran University of Science & Technology
emousavi@comp.iust.ac.ir

Abstract

Most applications are consisted of several activities that are fulfilled by different processes. And even processes are included different child processes named light processes or threads. The basic idea of dividing the whole activities to processes is followed by the reusability and sharing ideas. Therefore, applications need an IPC mechanism to establish the communication between the processes. Inter process communication that is known as IPC is a collection of mechanisms that meet the communication requirements between processes. System V defines standard for IPC mechanism named SVIPC. Different operating systems implement SVIPC standard in different manner. Therefore programs that are using the IPC mechanism have different structure in other operating systems. On the other hand reproducing program for various operating systems is a time consuming activity. Porting is a solution to writing programs with the least changes to port them on different operating systems. In this survey we present a brief introduction of various IPC mechanisms in the two operating systems and describe porting Windows' programs to Linux by mapping the IPC primitives as a solution. We present the porting as a solution to portable IPC programming. While the program is written with windows IPC mechanism can use our wrapper to be able to run in Linux operating system.

Keywords: Inter process communication, SVIPC, Linux, Windows, and Portability.

1. Introduction

One of the most important subjects in the programming is communication between the programs. This idea enables developers to program on top of the existing IPC mechanisms and uses their functionalities. But what is IPC? Inter process Communication (IPC) is a collection of mechanisms that provide communication between two processes with their different memory space. The point to consider is the unawareness of programs from each other. The independency causes programs are written in different domain and communicate easily. As a result IPC enable applications to have concurrent task. [1]

The only problem is the lake of uniform solution because different operating systems implement these mechanisms on their own way like Apple Event technology by Apple or Object Linking and Embedding by Microsoft and Message Passing by Linux. Some of them are common in all operating system like Pipe, but implementation mechanisms are different and cause to not be portable between operating systems.

Different communication techniques like message passing or shared memory are used to implement communication. On the other hand different implementation on various operating systems causes those written programs for a family of operating system cannot simply run on another family. Therefore, considering portability feature can regulate the problem. The objective of the following solution is to suggest a solution to implement portable IPC in heterogeneous environment with different operating systems without programmer effort to change and modify the program.

978-0-7695-3369-8/08 $25.00 © 2008 IEEE
DOI 10.1109/ADVCOMP.2008.38

Portability requirement is augmented with the growth in high performance computing applications and causes to increase various type of migration. And establish to programmer a safe programming while programmer doesn't need to change the program from one environment to another environment. It includes different methods and implementations for specific domains with individual requirements in, other words it requires ad hoc solutions [3].

Also portability is treated as a quality factor in software engineering [8, 9]. Unfortunately there is not a systematic way to identify the portability problems.
Portability ability of the programming language is a key point to generate portable programs or to implement a solution to port programs. Obviously, portability achievement with low level languages is more accessible, i.e. more near to machine code more flexibility to port the program. However, with the very low level languages, the simplicity of use is lost. In addition chosen language should be understandable by most machines like FORTRAN, COBOL or C. the languages other than C despite of standards have vast differences on different operating systems [6]. Languages like C, C++, Shell, Perl, Tcl, Python, Java and Emacs Lisp are highly portable. Despite the possibility to port the C code between UNIX species, C binding differences on other operating systems are the source of portability problems, although Windows NT supports ANSI/POSIX C API [12].
C language is suitable one because has capabilities like typedef that isolate the definition of machine dependent data types, sizeof(type) construct to find the size of any of these defined data types and the union declaration to define the overlaying of data [5]. On the other hand C is not fully type-safe language and that causes difficulties to declare portability on machines.
All these portable languages have to use operating system s system calls or API s for IPC mechanisms and extremely are depended to operating system. Operating systems have different architectures.

2. MS-WIN Architecture

In the first glance to the architecture of the MS-Win, it is consists of two major parts, kernel and user modes (Fig 1). And the only way to access the kernel objects is the Win32-API or other subsystems such as POSIX. The Win32 allows developers write applications that run on all versions of Windows. On the other hand POSIX provides a portable and standard-compliant environment to run an application on different operating systems [7]. In this area our focuses is IPC in the Win32-API and the equivalent mechanisms in the Linux.

3. IPC in MS-WIN

MS-Win provide a wide range of IPC mechanisms such as Clipboard, Data Copy, Winsock and others to send and receive the messages between two processes or copy the data in a shared section in memory or file. In the following nine IPC mechanisms supported by MS-Win are mentioned briefly.
In Clipboard mechanism coupling between the processes is low and the drawback is that the processes must agree on the data format. It can be used between applications and within an application. A memory object can be any data format (clipboard format). By Data Copy the data is being copied to the destination memory space from the source memory space after that communication is established by sending request message (WM_COPYDATA) to the destination process and identification of the sender by receiver. Dynamic Data Exchange does not define any restriction on the exchanged data format as previously mentioned one.
In contrary to the Clipboard that is one-time response to user command the DDE is a continual data exchange. It can be used between tight coupled application by customize the DDE data format to implement special purpose IPC. In the File Mapping the contents of a file can be mapped to a virtual memory. Mailslot enable processes to communicate each other in a one-way channel. Server is the process that creates the mailslot and mailslot clients by writing messages to their mailslots can communicate with the server. It is suitable one for asynchronous communication. Pipes are supported about in all operating systems. MS-Win supports two types of pipes, named and anonymous. The later one enable the related processes like the child process to exchange the data with their parents. To provide a duplex operation the processes need to create two anonymous pipes, one from client to server and the other one for the reverse direction. In contrary to the anonymous, the named pipes do

not have any restriction in the communication between non-related processors. A named pipe server is identified by the name of the pipe (well named pipe) and data will be transferred after both the server and the client connect to named pipe. Winsock is the MS-Win implementation of sockets popularized by Berkeley Software Distribution (BSD). It uses the communication capabilities of the underlying protocol. Therefore it provides a protocol-independent interface such that an application using Winsock can communicate with other applications using different socket implementation on other systems. It is very helpful to implement the communication between heterogeneous operation systems. Remote Procedure Call enables applications to call functions of the other applications remotely. The RPC facility provided by Windows is compliant with the Open Software Foundation (OSF) Distributed Computing Environment (DCE). Therefore applications on different operating systems that support DCE can communicate each other. Since the RPC handle communication between different hardware architecture, it has to support automatic data conversion [8].

It has to be mentioned that the only Data Copy, Pipe Anonymous and File Mapping do not support remote communication and the others can be used in the remote environments.

4. Linux Architecture

Unlike Windows, Linux does not have subsystems in the user mode. Kernel resources like IPC mechanisms are available for developers to implement communication between processes [7].

5. IPC in Linux

Linux provides different kind of Inter process communication mechanisms and some of them are the same as MS-Windows does. For example they are included RPC, message queue, pipe, socket and IPC (semaphore and shared memory).

Pipe mechanism is divided to three types, half/full duplex and named pipe. Half duplex is a one way connection of the standard output to the standard input of different processes. It is most beneficial in the creation of child processes in multi process communication because it inherits file descriptors from its parent. Named pipe does not have the communication limitation between parent and child. And it is a special file in the file system. So it can be reused after doing all I/O operations. Full duplex pipe is a bidirectional stream based connection. System V IPC introduces three communication mechanisms (message queues, semaphores, and shared memory). To use the System V IPC it is sufficient to obtain the unique id of the IPC object. Message queue is a linked list in the address space of the kernel. On the other hand semaphore is an integer to lock the shared resources in a multiple process environment. As a matter of fact message queue is a linked list within the address space of the kernel. Messages are sent to the queue in order and can be retrieved in different ways. Semaphore acts as a counter and is used by processes to control access to shared resources. For example to prevent writing to a memory segment in shared memory. It implements locking mechanism. For example, to prevent writing to a same memory segment more than more than one in shared memory mechanism. A mapped and shared area of memory by multiple processes is called shared memory. A process can create a segment and share it by other processes [9].

6. Related Works

Portable inter process communication enable different processes on a network establish communication without restriction to the physical location type of family of operating systems. Therefore porting part of software or a module between different operating systems is possible by communicating processes.

Although most of the operating systems provide IPC primitives, an IPC base application cannot be simply ported because of lack of standards in their implementations. In addition each implementation has own particular objectives that are not generic. IPC mechanisms establish communication between processes locally or remotely. There are a lot of researches to develop portable local IPC mechanism or remote IPC mechanism.

Among them we can mention Adaptive Communication Environment (ACE), ACE provides an object oriented framework for developing concurrent communication software. The ACE wrappers are at the application level and don not say anything about low level system call mapping [13].

Portable IPC that is implemented on Vanilla UNIX, consider that all the processes in a tree have an ancestor to enable communication

between the processes that are not parent and child by pipe mechanism. The main objective of that is to help clients IPC channels on servers [11].

Sun Microsystems developed ONC RPC as part of the Network File System project to leverage the RPC in heterogeneous environment. As a matter of fact ONC RPC designed for network programming and distributed system. ONC RPC enable modules call function of other modules on different operating systems. Because it is based on the RPC mechanism, the programs communication is tightly dependent on port number. Porting programs on a different operating system especially between two most famous operating systems, UNIX and MSWindows is an interesting topic. Wine and Cygwin are the two instances of the porting algorithms. In 1993 the Wine project has been started to run Win3.1 programs on UNIX and now it is capable to run Win32 programs. Wine is a translation layer that can run windows programs on POSIX compatible operating systems. Basically, Wine implements the windows system calls on the UNIX. Wine does not support different IPC mechanisms. Programs can use DDE and Socket with limitation. Cygwin is collection of the software tools to simulate the Windows on the POSIX compatible operating systems like UNIX, BSD or LINUX. Unlike the Wine, Cygwin recompile the program by using the header files. Cygwin was started in 1995 by Steve Chamberlin when he found WinNT uses the COFF. It is reimplementation of the GCC to generate the Windows programs.

Mono and .Net are other instances. They provide platforms to port the applications from Windows to LINUX specially written based on .Net framework. They like Java Virtual Machine do not enable developer to access the low level system calls so that they are complex for programming for clusters [14].

7. Wrapper

As a matter of fact software components are created with different purposes, and the common sense is using existence component capabilities although components have different internal technology. Wrapper is software pattern to leverage the capabilities of component, object or even a library without involving the internal complexities of the used component. In this scenario the wrapper is used to implement the compatibility between IPC mechanisms in two different operating systems in one side, and the simplicity of the programming model for developer on the other side. The wrapper implementation can be done in different layers like kernel, application and user. Obviously, the kernel level implementation needs the modification of the kernel that is not our concern because of the closed source operating systems do not allow modifying the kernel like open source operating systems simply. On the other hand the application level implementation depends on the type and the requirements of the specific application. The user level implementation of wrapper provides the transparency. And it provides a mechanism that can be tailored to any unpredictable requirements. The wrapper mechanism presented here unlike the normal behavior of wrappers hide the incompatibility between the IPC mechanisms in two operating systems. Therefore the written programs by using the wrapper in the Windows environment do not need to modify their code to be compatible with the Linux.

In the absence of the wrapper, process continues the normal behavior and invokes system calls to perform local and remote IPCs. The local system, for example MS-Win, has responsibility to establish the Inter process communication (Fig 4).

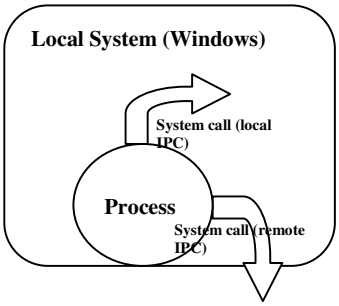

Fig 4- process in normal behavior

In case the process with the wrapper linked, the system calls requested by the process is redirected to the wrapper, then the remote inter process communication is established by the wrapper between the local process and the remote process on the different environments (Fig 5). Data conversion, IPC compatibility and diagnosing IPC mechanism dynamically, are the wrapper responsibilities.

The Wrapper consists of header files which enable the programs especially on closed source operating systems to communicate with the open source operating systems like Linux.

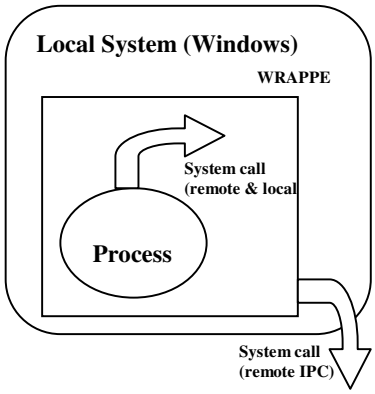

Fig 5- process in conjunction with Wrapper

With Wrapper, the program base on windows pipe convert to a program base on Linux pipe and can be run in Linux environment and wrapper transmit the program to Linux

windows	Linux
CallNamedPipe()	open()
ConnectNamedPipe()	
CreateNamedPipe()	mknod()
DisconnectNamedPipe()	close()
GetNamedPipeHandleState()	
GetNamedPipeInfo()	
PeekNamedPipe()	
SetNamedPipeHandleState()	BLOCK
TransactNamedPipe()	
WaitNamedPipe()	
ReadFile()	read()
WriteFile()	write()

Table 1: Pipe mechanism mapping

environment and after run it, the result transmit to windows.

The wrapper module is written in C++ language compatible with the WIN32-API on MSWin and in C language compatible with System V API on LINUX. The communication model is base on common remote IPC like RPC, pipe or socket, therefore it is not restricted to communication between MSWin and LINUX and it can be deployed on a cluster included pure MSWin operating system. The most important objective of the wrapper module is simplicity of usage. It is enough to add the wrapper like any other header files to the source without any changes to the source. This feature enables developers to use the existing programming model.

7.1 Wrapper s Structure

In the proposed solution we chose MSWin and LINUX operation systems. Mapping analysis of system calls between the two operating systems is required as the first step of porting.

We can divide the system calls to two types. The first one can be mapped to each other by parameters, returned type and such. The second one depends on the context of the operating system and they may map or not, because for some functionality there are more than one equivalent and for some there is not any [10].

Pipe mechanism is supported by the two operating systems. Table 1 shows the equivalency of some functions of the named pipe in the two operating systems.

Named pipe in LINUX acts like as a regular pipe and is a kind of FIFO with these differences [9]: 1.Exist as a device special file in the file system.2.Different ancestry processes can share data through a named pipe. 3. The named pipe remains in the file system for later use after all I/O. and in the Windows have the following characteristics [8]. 4. One-way or duplex pipe, one for server and another for client. 5. While each instance of the name pipe has its buffer and handle they share the same named pipe. 6. Multiple clients can use the same named pipe simultaneously.

There is not one to one equivalency between the implementation in two operating systems for pipe mechanism. Considering the first characteristic of the named pipe, functions that do not have any equivalent can be simulate to combination of function on files like blocking equal to SetNamedPipeHandleState() or obtaining information equal to GetNamedPipeInfo().

7.2 Wrapper s Features

The following features are thought in the requirement analysis phase and have been considered in the structure design phase:

- The least changes in the existing programs or even without changes.
- Compatibility with the MS-Win architecture which provide the high level performance.
- Deployment on the Linux cluster, Windows cluster or combination of both of them (x86, x64, IA64).
- Robust and reliable in the industrial domain.
- Simplicity in configuration and installation.

• Scalability to support thousands of the processes.
• Integrating to system resource management and analysis tools.
• Run by the wide range of C/C++ compilers and debuggers.

8. Conclusion

The wrapper s solution followed the portability of IPC base processes and converts programs base on Windows pipe IPC to Linux pipe IPC and transmits the program in Linux environment and after run, returns the results to Windows. Wrapper project s important purpose is easy of programming in IPC programming and firstly chose pipe inter process mechanism. Wrapper project want to extend to all IPC mechanism and convert all kind of them to each other.

9. References

[1] L. Lamport, "*On Interprocess Communication*", Springer-Verlag, 1986.

[2] J. D. Mooney, "*Bringing Portability to the Software Proocess*", The Department of Computer Science and Electrical Engineering at West Virginia University, 2001.

[3] S.R. Schach,"Classical and Object-Oriented Software Engineering (3rd ed.)"Richard D. Irwin, Chicago IL, 1996.

[4] I. Sommerville, "Software Engineering (5th ed.)" Addison-Wesley, Reading MA, 1996.

[5] T. L. Lyion, *"Inter-Unix Portability",* in B .A. Tague, editor, "C Language Portability", Bell Labs Internal Memorandum, September 1977.

[6] S. C. Johnson, D. M. Ritchie, "Portability of C Programs and the UNIX System*", The Bell System Technical Journal, Vol. 57, No. 6, Part 2, July-August 1978, pp 2021-2048.

[7] L. Twork, L. Mead, B. Howison, JD Hicks, L. Brodnax, J. McMicking, R. Sakthivel, D. Holder, J. Collins, B. Loeffler, "UNIX Application Migration Guide", Chapter 2, Microsoft Corporation, October 2002.

[8] Microsoft Developer Network, "Interprocess Communications", http://msdn.microsoft.com/en-us/library/aa365574(VS.85).aspx

[9] S. Goldt, S. van der Meer, S. Burkett, M. Welsh, "The Linux Programmer's Guide", Version 0.4, March 1995

[10] S. S. Muthuswamy, K. Varadarajan, "*Port Windows IPC apps to Linux*", IBM, 14 Apr 2005.

[11] M. Rain, "*Portable IPC on Vanilla Unix*", ACM Publisher, Volume 24 , Issue 5, May 1989

[12] E. S. Raymond, "The Art of Unix Programming", Copyright © 2003

[13] S. D. Huston, J. CE. Johnson, U. Syyid, "ACE Programming's Guide, The: Practical Design Patterns For Network and Systems Programming", Addison Wesely, Nov 14, 2003

[14] A. Merta, "*MONO: an alternative for the .NET framework"*, Software 2.0 magazine, February 2005.