JACKY Operating System

Suvankar Barai

Abstract— JACKY is a new operating system kernel for IBM-PC compatible computers. This kernel is small, fast, and open source. It operates exclusively in 32-bit mode and features real preemptive multitasking and virtual memory. An operating system kernel is a big enough challenge to be discouraging at times. As an example, the command Interpreter I wrote over an one year period is trivial by comparison. On the other hand, since I do everything by myself I am able to keep the development on a unified path. The architecture that develops is -- I hope -- consistent (for better or worse) and thus the end product reflects the vision of a single programmer. It can be argued that this is the good, old-fashioned way of producing software.

Index Terms— Stability, reliability, and security, Native Graphical environment, Command-line capabilities, Networkable, Compatible.

I. INTRODUCTION

JACKY (Operating System) is a home-brewed computer operating system kernel, which is still -- and is continually -- in development. JACKY has initially been designed to support PC-compatible computers. The design and interface philosophies of JACKY are driven by the fundamental goal of keeping some of the best features of other successful systems, while discarding many of their notorious weaknesses. It follows then, that however many ideas JACKY borrows from other operating environments, it is not -- and does not try to be -- entirely compatible with any of those other systems. Although some aspects of JACKY will probably seem familiar with any other operating environment.

Some of the higher-level conceptual goals are as follows:

1. Stability, reliability, and security. These are primary objectives. Not always achieved, but always important.

2. "Native" Graphical environment

- The base-level graphics server (analogous to an 'X' server in Unix, but not X) will be integrated into the kernel. A default GUI shell environment must load and run "straight out of the box", with no complicated setup procedure.
- The interface must be trivial to learn and use, even for a computer novice. Most of its elements should be familiar to all computer users. Just like the old computer-industry cliche: "Something my Mom can use".
- Something which might seem like a contradiction of the previous item: A new metaphor for the GUI environment. The tried-and-true desktop/office/files/folders paradigm is becoming a little bit dated. This interface will put a new spin on graphical shell design, without making it unfamiliar or non-intuitive. Stay tuned for details.
- To the greatest extent possible, the user should be able to perform all tasks, including administrative ones, using this "point-and-click" interface -- no need to edit mysterious configuration files by hand.

3. Powerful command-line capabilities (text windows and scripting):

- Users must be given the ability to operate in a text-based environment if they prefer to do so.
- Existing popular command shells will be supported. The native command shell will contain features found in various systems (Unix and non-Unix).
- Most common text-mode commands available on other popular systems will be included and will, as much as possible, be use model compatible.
- Some of GNU's Unix-like tools will be available (see http://www.gnu.org).
- To the greatest extent possible, the user should be able to perform all tasks, including administrative ones, using this text interface. Configuring mysterious configuration files by hand is, therefore, optional.

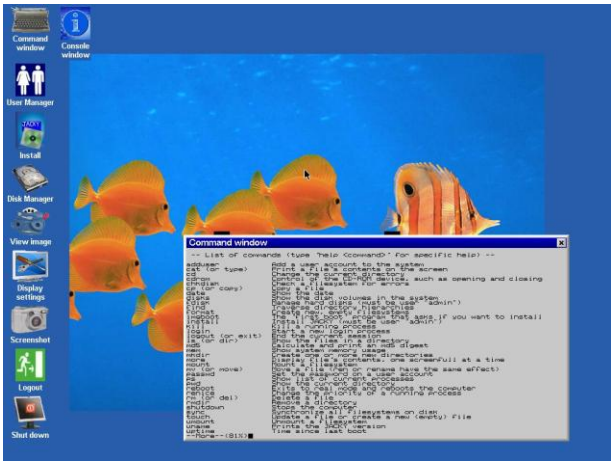

Fig. 1 Command-line in Graphical Environment

4. Highly networkable. JACKY will be very network oriented. Stay tuned for more details about this as well.

5. Highly compatible. JACKY will conform to existing

standards to the greatest extent possible. It is not desirable for JACKY to define new formats (such as a new filesystem type).

Examples of supported standards will include:

- Filesystem types
- Executable/object/library file formats
- Image, sound, font, compression and (enhanced) text file formats
- Encryption algorithms
- Network protocols
- Software development environment conventions
- Operating system API calls.
- Hardware interface standards (e.g. VESA)
- Basic POSIX compliance, where possible

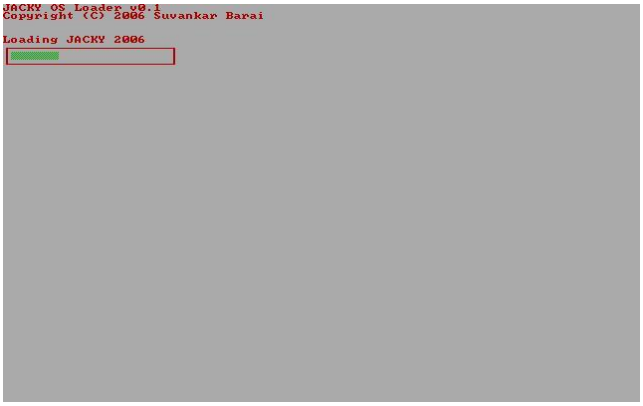

Fig. 2 Loading File-system and Kernel

II. CURRENT DEVELOPMENT STATUS

JACKY is still in a fairly early stage of development. Coding work was begun as a part- time operation in late 2005. The majority of the code is written in C, with portions written in x86 Assembly Language. The present incarnation of JACKY boasts enough features to classify it a promising project, but not enough to make it useful to non- technical users in its current state. The following is a list of implemented and unimplemented functionality; keep in mind that this does not represent the complete list of planned features -- only short- and medium- term goals are listed here:

Implemented (or mostly implemented):

- Fully 32 bits, "protected" mode
- Preemptive multitasking and multi- threading
- Virtual memory, and memory protection
- Flat linear memory management
- Graceful processor fault and exception handling
- Good random number capability
- Filesystem support for: Virtual filesystem API Buffered, asynchronous filesystem IO Multi- threaded filesystem functions and device locking 12, 16, and 32- bit FAT filesystems (commonly used by DOS and Windows) Multiple mount points, multiple disk slices, or partitions, on a single device
- Abstracted loading and management of device drivers
- Delayed event scheduling
- Device support for Single 486, K6/MII/Pentium processor (or better) RAM above 64Mb Programmable Interrupt Controller (PIC) System timer chip Real- Time Clock (RTC) chip Keyboard controller Text console IO Direct Memory Access (DMA) controller Floppy disk drives Fixed (hard) disk drives.

Partially implemented:

- Kernel API
- Native command line shell
- Loader
- Standard C library

Unimplemented (or mostly unimplemented):

- Multi-user operation
- Inter- Process Communications (IPC) facility
- IO Protection
- FPU state saves
- Assembler and compiler (ports of NASM/GAS and gcc)
- Native linker, or satisfactory port of GNU linker, ld
- Support for Elf, a.out, and PE executable formats, loader relocation
- Dynamically loaded/linked libraries
- Emulation of other operating systems' APIs
- Filesystem support for: Ext2/Ext3 filesystems (commonly used by Linux) NTFS filesystems (commonly used by Windows NT/2000 and Linux) CD-ROM filesystems (ISO9660/Joliet) (others, as demand dictates)
- Device support for: Multiple processors 3DNow! and MMX processor extension Plug and play PCI bus devices Modem Network Interface Card (NIC) Printers (many others)

III. DEVELOPMENT ENVIRONMENT

JACKY is developed under Red Hat Linux 9.x, using the GNU C compiler and the NASM assembler.

IV. ACKNOWLEDGEMENTS

This work was supported by Jadavpur University, Kolkata.
Website: www.jackyos.co.nr

V. REFERENCES

[1] Katrin Becker (becker@cpsc.ucalgary.ca) at the University of Calgary for helpful advice about free- list management in filesystems. I should have paid more attention in class.
[2] John Fine (johnfine@erols.com), Alexei A. Founze(alex.fru@mtu-net.ru), and the rest of the regular contributors to the comp.lang.asm.x86 and alt.os.development newsgroups. Thanks for always taking the time to help people.
[3] Jerry Coffin (jcoffin@taeus.com) and Ratko Tomic for

posting information about alternate text mode video configurations.
[4] Ralf Brown (ralf@pobox.com) for creating and maintaining the indispensable Interrupt List (http://www.cs.cmu.edu/afs/cs/user/ralf/pub/WWW/files.html); David Jurgens for HelpPC. Both are excellent PC programming resources.
[5] Frank van Gilluwe for "The Undocumented PC" (Addison-Wesley, ISBN# 0-201-47950-8); Tom Shanley for "Protected Mode Software Architecture" (Addison-Wesley/Mindshare ISBN# 0-201-55447-X)
[6] Bob Watson (rhwatson@sympatico.ca) for maintaining the MS-DOS7 Commands page at http://www3.sympatico.ca/rhwatson/dos7/.
DOS/Windows installation tools wouldn't work as well as they do without the aid of this site.