

# OaSis: An Application Specific Operating System for an Embedded Environment

Gurashish Singh Brar  
Susmit Biswas

Sudipta Kundu<sup>\*</sup>  
Arijit Mukhopadhyay

Pratik Worah  
Anupam Basu

*Dept. of Computer Science and Engineering, IIT Kharagpur*

*<sup>\*</sup>Dept. of Mathematics, IIT Kharagpur*

## Abstract

*Portable and affordable assistive devices are calling necessities for the physically challenged population putting forth the requirement of designing embedded systems tuned to specific assistive applications. In this paper, we present a low footprint embedded operating system, OaSis, which forms the core of embedded devices being developed for the speech impaired, the visually impaired and those with neuro-motor disorders. The target hardware, housing OaSis and on which the applications are ported is Geode GX1-233 processor based POS-563 of Advantech. The system is presently being ported to more energy efficient ARM based platform. This system currently supports three applications: Shruti – a Vernacular speech interface, Sanyog – a Visual Communication interface and a web browser. The OaSis embedded operating system has been optimized to 90 KB and provides all the features required for the specific applications. The different modules of OaSis have been described and have been compared with other operating systems.*

## 1. Introduction

Assistive Technology [1, 5] deals with development of hardware and software that enables physically challenged people to overcome the hindrances in efficient communication. Many of the modern assistive devices are computer based [2, 3]. However, for better usage, it is required that the systems be on low cost portable platforms. The affordability issue is all the more important for the people in developing countries, since most of the portable systems are costly with respect to these target users. The portability issue therefore calls for indigenous development of low cost embedded assistive systems.

The design or selection of suitable operating system kernel forms a basic need of such embedded system design. This paper presents OaSis, a low footprint kernel. The hardware platform, on which the kernel and the applications were first developed and ported, is aGeode processor based Advantech board. The applications that have been developed around OaSis, include a text-to-speech system to be used by the visually handicapped -Shruti, a visual communication system to be used as an Augmentative and Alternative Communication System (AAC) [4, 5, 6] for use of the people affected with cerebral palsy - Sanyog. In addition, a Web Browser is included for internet accessibility.

This paper concentrates on the design of OaSis and also presents some of the embedded applications developed around OaSis.

The paper is organized as follows.

Section 2 presents the overview of the target hardware platform and the kernel of the OaSis operating system. The next section discusses the application programming interface (API) provided with OaSis. Section 4 describes the device drivers. Section 5 discusses the optimization of OaSis. Section 6 presents a comparison of OaSis with other operating systems. Section 7 describes the ported applications followed by the concluding section.

## 2. Overview of OaSis and the Hardware Platform

The aim of OaSis is to provide an optimized low footprint operating system. The current system consists of the OaSis nanokernel, a modified form of the uIP TCP/IP stack [8], emulation API's providing the low-level system functionality required by the applications and drivers for the following: clock, ATAPI, Ethernet, LCD, keyboard and sound blaster.

## 2.1. The Hardware Platform - POS – 563 Port

**2.1.1. The POS – 563.** The POS-563 is a low cost, fan-less Geode GX1-233 board specially designed for POS applications. The GX1-233 processor allows for fan-less operation that virtually eliminates heat buildup problems. The POS-563 has one PCI/ISA expansion slot, four digital I/Os and four on-board serial ports each with +5 V/+12 V power.

**2.1.2. The POS – 563 Port.** In preparation for working with real POS–563 hardware, a port of the Geode GX1-233 was made using 32-bit GNU C compiler and GNU inline assembler.

## 2.2. OaSis Nanokernel

The OaSis kernel has most of the characteristics of a standard embedded OS kernel i.e. interrupt support and soft Real-Time scheduling. The scheduler and the interrupt control are closely interwoven. The data structures are highly optimized to allow the full kernel to reside in memory. OaSis, a modified subset of Montague's JN nanokernel [7] follows a classic soft-real-time architecture informally known as a Cutler kernel. The present kernel consists of a single work-loop, driven by a queue of control blocks. It begins execution in response to an interrupt. Once started, it continues to execute until no control blocks remain in the queue. Each control block contains the address of a routine which the kernel must execute. Historically, these routines have been called *fork* routines in the Cutler kernels. Since fork routines cannot block and must have a bounded execution time, they have many characteristics of routines written for a hard-real-time environment. Neither the fork routines nor the kernel need to use explicit mutual exclusion to access global data structures, reducing the need for explicit synchronization. The OaSis currently supports Ext2 file system [9]. For networking the OaSis supports Ethernet [11]. The full kernel architecture was not implemented. The subset is referred to as a *nanokernel*. Although the term has been disparaged, it is believed to be warranted [13].

The current image size of OaSis is well within 90KB including all the modules i.e. the nanokernel itself, the various application interfaces and the different device drivers, excluding the speech database required for Shruti.

## 3. The Application Programming Interface (API)

This section documents the API required by the application to execute on OaSis. The API is divided into

4 classes: thread, graphics, file and inter process communication.

### 3.1. Thread APIs

The threading API were suitably modified from the JN [10]. Thread priorities range from 0 to 16. This is required for a soft Real Time environment to provide the developer with a larger range, so as to effectively decide the priorities of his applications. A *sysThreadCreate()* API routine creates a thread in suspended state, *sysThreadExit()* can terminate any thread, and *sysThreadYield()* rotates the threads at the current priority level. Any thread can be removed from scheduling consideration by *sysThreadSuspend()* and restored to scheduling eligibility by *sysThreadResume()*. The unique integer thread ID of the caller can be obtained as well as the priority of any thread can be set and read.

### 3.2. File APIs

Though the file system used by the OaSis embedded operating system is a subset of the Ext2 file system, all the standard conventions have been followed and the standard data structure for the *inode* and the *file descriptor table* has been implemented. The *superblock* structure also remains the same. Some changes have been made because of the application specific nature of our system.

Shruti produces the output speech file by concatenating small wav files, read from a database. So to make Shruti run in an optimized way, OaSis needs to search through the database very quickly and also read from and write to a file quickly. The same requirement exists for Sanyog. OaSis thus uses a binary search strategy when searching for files within a directory to quicken the search. As far as the API functions are concerned there is the standard *open*, *close*, *read*, *write* and *lseek* functions, implemented keeping in mind the application's requirement.

### 3.3. Graphics APIs

The Sanyog Application needs a grid type layout for displaying its icons. A user will select a set of icons from the display with switches attached to the serial port. Displaying graphics in a fast and optimized way was a priority. The graphics API has a simple Java AWT type layout manager. The API consists of the *TextClass*, *GraphicsClass*, *LayoutBoxClass* and *GridLayoutClass*. The *GridLayoutClass* extends the *LayoutBoxClass* and divides the display area into grids, where each grid box can contain a graphics object or text object or both. The *LayoutBoxClass* takes into account

the graphics and text object attached to the box, and displays the object in the grid box whenever the display\_object function is called. The TextClass and GraphicsClass read and buffer the text and graphics object respectively. The system currently can display portable pixel map (PPM) format files.

### 3.4. Inter Process Communication (IPC) APIs

The Web Browser application downloads a web page from the internet and then applies a transcoding algorithm to convert the page, because of the limited display capability of the client device which is then passed to Shruti. Inter process communication is required in OaSis between Shruti and the browser and between Sanyog and Shruti. So a Pipe message passing data structure was created.

### 4. Drivers

Device drivers for the serial UART, the Keyboard, Sound Blaster and Ethernet have been implemented as described below:

- Realtek Ethernet: This driver has been implemented for the Web Browser application. This driver has two major components:
  - a) Packet Transmission: The transmit path of RTL8139 uses four descriptors, each descriptor having a fixed IO address offset. The four descriptors are used in round-robin fashion. Early transmit threshold is also specified in the descriptor.
  - b) Packet Reception: The receive path of RTL8139 is designed as a ring buffer. Data coming from line is first stored in a Receive FIFO in the chip, and then moved to the receive buffer when the early receive threshold is met. The status of receiving a packet is stored in front of the packet (packet header).
- Sound blaster: The need for this driver arose due to the text to speech application Shruti. The driver uses DMA to transfer the sound file to a buffer where from the DSP chip processes it to get the sound.
- Keyboard: This driver is used by all applications. Implementation uses the standard approach i.e. with a circular queue and a process queue for the device.
- Serial UART: The need for this driver came because in the iconic application, Sanyog a user can give input to the application through special access system as described in Section 7.1.

### 5. Optimization of OaSis:

OaSis has been optimized in many ways. The thread, work blocks and memory data structures have been highly optimized because only a few applications will be running at the same time. Light thread data structures allow rapid scheduling. Only necessary drivers have been implemented. The file system and the file APIs have been modified. A binary search strategy has been implemented to facilitate a rapid database search for the TTS application. Sanyog uses a grid type interface to display the icons and also requires displaying the same icon more than once at times. The Graphics API has been written considering the display constraints of an embedded device. The Graphics API has very simple data structures in which it reads a graphics object and buffers it. This helps in a much faster display of icons.

### 6. Comparison

In this section we present a comparison of OaSis with respect to other standard operating systems.

#### 6.1. System size

Table 1 gives a comparison of the storage requirements for various embedded operating systems with the OaSis operating system which clearly shows the difference in size achieved by OaSis.

Table 1. Storage Requirements

OaSis	Embedded XP	Embedded Linux	CE.NE T	Micro Linux
90KB	100MB	16MB	16MB	512 KB

#### 6.2. Context Switch Time

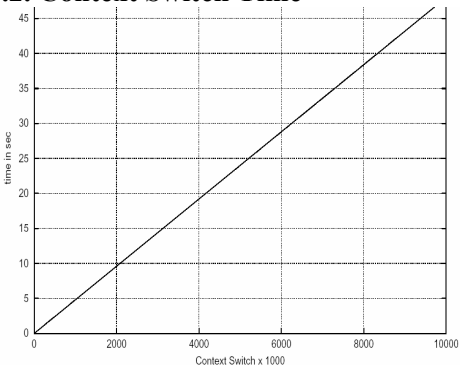


Figure 1. Time in sec (y axis) vs. context switches x 10000 (x axis).

The values for context switch time for OaSis were obtained for a set of discrete points and the near linear relationship obtained is plotted in Figure 1. Operating systems like RH - 7.2, Windows XP, Windows 2000 SP2 have a context switching time which varies with the load and ranges from a minimum of 3  $\mu$ sec for Win 2K SP2 and maximum of 14  $\mu$ sec for Red Hat 7.2 Linux. This data was collected on a TP600X, 650 MHz by Bradford [12]. In comparison, our system has a context switching time of approximately 5  $\mu$ sec, which seems pretty good for our embedded applications.

## 7. Applications.

### 7.1. Sanyog – Visual Communication Interface

The iconic communication application[14] provides facilities physically challenged people to communicate normally. The application has a very simple GUI through which user selects icons. It has the facility to form natural language sentences with various moods, tenses and also interrogative sentences. The application can be integrated with special purpose input devices specially fabricated for people afflicted with motor-neuron disorders. It is also equipped with speech synthesis technique.

### 7.2. Shruti – Vernacular Speech Interface

The Text to Speech (TTS)[15] application acts as a computer interface for the visually challenged, for whom graphical interfaces are not viable also serving the purpose of the voice of the speech impaired. The TTS is based on a concatenation of speech approach. Given a sentence in text, the synthesized speech will be generated after some steps.

### 7.3. Web Browser

The Web Browser is a very low footprint simple embedded browser. It uses a set of routines which take input from a user, use TCP/IP to perform Web transactions, and send the results to an output device, the LCD display. The application can also make direct calls (calls not initiated by an input device) to the browser to perform such actions as loading a specific HTML file.

## 8. Conclusion

A working network computer with this custom operating system has been implemented on a Geode GX1-233 processor based Advantech POS-563 board. While this exercise served the purpose of proof of concept, we are now porting it to ARM processor based platforms to

result in power efficient portable system. The paper describes a novel operating system so that the specified applications would perform better in this custom OS than in a standard OS. The system is reasonably robust and can serve as a test bed for future work. It was shown that it is possible to do TCP/IP network development and related research at a very low cost. The applications have been ported and their successful execution proves the efficacy of OaSis.

## 9. References

- [1]. Proceedings of the *ACM SIGCAPH Conference on Assistive Technologies*, 2000.
- [2]. Newell Morris A, Booth L, and Ricketts I, Syntax PAL: A writing aid for language-impaired users, in *ISAAC-92*. [Also in *Augmentative and Alternative Communication*, Vol.8, 1992].
- [3]. Pennington C.A. and McCoy K.F, Providing Intelligent Language Feedback for Augmentative Communication Users, in *Assistive Technology and Artificial Intelligence, Applications in Robotics, User Interfaces, and Natural Language Processing*, (Mittal V.O et al eds.), *LNAI 1458*, Springer Verlag, 1998
- [4]. K. Rumble Gilian and Larcher Janet, *AAC Device Review*, Vocation Pub. 1998
- [5]. Vanderheyden P B. and Pennington C.A, An augmentative communication interface based on conversational schemata, in *Assistive Technology and Artificial Intelligence, Applications in Robotics, User Interfaces, and Natural Language Processing*, (Mittal V.O et al eds.), *LNAI 1458*, Springer Verlag, 1998.
- [6]. Warrick A and Kaul Sudha, *Their Manner of Speaking*, Indian Institute of Cerebral Palsy, Kolkata, India, 1997
- [7]. B. R. Montague. *JN: OS for an embedded Java network computer*. *IEEE Micro*, 17(3):54-60, May/June, 1997.
- [8]. Adam Dunkels. *uIP - A Free Small TCP/IP Stack*. Technical report, <http://dunkels.com/adam/uip/>.
- [9]. Rémy Card, Theodore Ts'o and Stephen Tweedie. *Design and Implementation of the Second Extended Filesystem*. The First Dutch International Symposium on Linux, ISBN 90-367-0385-9.
- [10]. B. R. Montague. *JN external API*. Technical report, U. of Calif. Santa Cruz, UCSC-CRL-97-17, 1997.
- [11]. Realtek Semiconductor Corporation. *Realtek 3.3V single chip fast Ethernet Controller with power management RTL8139C (L)*. Revision 1.1, November 1999.
- [12]. Dr Edward Bradford. *High performance programming techniques on linux and windows*. <http://www-106.ibm.com/developerworks/linux/library/l-rt9/>. July 1, 2002.
- [13]. Jochen. Liedtke. *Towards real microkernels*. *Communications of the ACM*, 39(9):70-77, September 1996.
- [14]. Basu A., Sarkar S., Chakraborty K., Bhattacharya S., Choudhury M. and Patel R. Vernacular Education and Communication Tool for the People with Multiple Disabilities. Presented in Development By Design Conference, Bangalore, 2002.
- [15]. Sen D., Sen S., Chakraborty S.J., Chakraborty A. and Basu A. An Indian Language Speech Interface for Empowering the Visually Handicapped. *Proceedings of International Workshop on Frontiers of Research in Speech and Music*, Kanpur, 2003. (pp 113 - 120)