7th International Scientific Conference
Technics and Informatics in Education
Faculty of Technical Sciences, Čačak, Serbia, 25-27th May 2018

Session 2: IT Education and Practice

UDC: 378:004.43

Possibility of Deploying COSMOS Operating System on Personal Computers

Stefan Đokić [1*], Stefan Stanojlović [1], Dejan Vujičić [1]
[1]University of Kragujevac, Faculty of Technical Sciences Čačak, Serbia
[*] sdjokic722@gmail.com

Abstract: *Knowing the concept and way of implementation of the operating systems is one of the most important tasks that computer experts should meet. Therefore, educational versions of operating systems are being designed and implemented. The development of such operating systems provides the opportunity for interested parties to become familiar with the possibilities of using various software tools for their realization. An example of such educational operating system is a set of program modules known as COSMOS. Modules are written in C# programming language and Visual Studio development environment is used for their translation and integration into a unique system.*

Keywords: *Operating systems; education; development environment; programming languages*

1. INTRODUCTION

The majority of computer users are familiar with only a few operating systems. For desktop computers, these are mostly MS Windows, Linux, and Mac OS [1]. On the other hand, in mobile devices, such as smart phones and tablets, Android, iOS, and Windows Mobile operating systems are primarily used [2], [3], [4]. However, this does not diminish efforts in which researchers are trying to come up with new operating system solutions. Particular emphasis should be placed on research in which operating systems are developed for educational purposes [5].

The operating system as the fundamental computer subsystem represents an important factor in the education of every computer expert. Therefore, it is necessary for students of Computer Science to learn about the functions of operating systems and the ways of their realization. A number of systems has been developed for this need. Some of them have the form of a simulator, while others can function as real operating systems. The main goal of such systems is to bring together students and everyone who wants to approach the principles of operation of the operating systems, their installation, and use in real terms. In practice, most operating systems for educational purposes are based on UNIX and its variant for microcomputers, the Linux operating system. Accordingly, there is a number of UNIX/Linux [6] distributions in practice, and the most common are MAX, MINIX/MINIX 3 [7], EdUbuntu, UberStudent Linux, SkoleLinux, EduLinux, and others.

A typical example of this system is COSMOS (C# Open Source Managed Operating System) operating system [8]. Since it is an open source program system, as well as integrated into Microsoft Visual Studio development environment, it provides great opportunities in learning about the implemented modules as well as in upgrading them. The aim of this paper is to reveal the possibilities of COSMOS operating system, as an educational tool, but also as a basis for developing a standalone operating system.

2. WHAT IS COSMOS OPERATING SYSTEM?

COSMOS is not a complete operating system, but a set of development modules that can form a new operating system. The development environment of this operating system is Microsoft Visual Studio. Basic COSMOS routines are mostly written in C# programming language. A minor part of the code is written using the assembly language X#. This programming language has been developed by Microsoft to make the development of COSMOS operating system easier. However, any programming language included in Visual Studio or .NET development environment can be used to further upgrade the COSMOS operating system.

Integration into Microsoft Visual Studio makes COSMOS significantly different from similar operating systems. As a result, the further development of the COSMOS operating system can be fully realized through Visual Studio. It is especially important that direct execution can be realized within Visual Studio. COSMOS is executed in a virtual environment that speeds up the development and testing of the operating system.

The basic virtual environment represents VMWare for ease of integration into the development project. In addition, other virtual environments, such as Bochs and VirtualPC, can be used. It is important to note that in addition to starting the operating system via a USB and a CD-ROM drive, booting can also be accomplished from a remote computer over the network, using the PXE (Preboot Execution Environment) technology [9].

COSMOS operating system has rings as its security feature [10]. There are four rings that differentiate areas of operation for adequate users. They prevent code in one ring from accessing fields in nonadjacent rings. The ring system in COSMOS operating system is represented in Fig. 1.

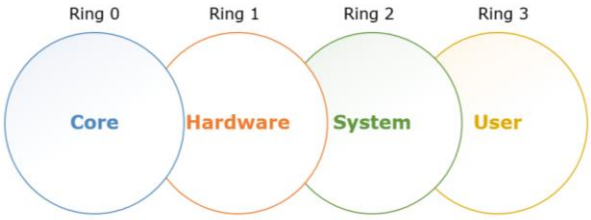

Figure 1. *Ring system in COSMOS operating system*

The bootloader that COSMOS is using is Syslinux [11]. It is used to boot COSMOS code in BIOS, and once it is done, the bootloader is removed from the memory. The filesystem COSMOS is using is FAT, but its implementation is still in progress.

Although COSMOS is using Visual Studio and .NET as its development environment, it cannot use their libraries. In order to run COSMOS on real hardware, all function calls that would normally use Windows API have to be replaced with COSMOS proprietary code. These codes are called plugs and they represent sections of code that hide assembly code running under the hood. They are C# class wrappers around assembly kernel handling code, but they can also be programmed in assembly or X#.

COSMOS appears in two types of distributions, DevKit and UserKit. DevKit is a version of the operating system developed directly by the COSMOS project team. Although DevKit contains the latest and most important features, there is a number of problems. However, there may be problems with the development of the operating system, and even the inability to start it up. On the other hand, UserKit is a stable version of COSMOS ready for installation. UserKit does not follow the development of DevKit because it is updated periodically. However, using UserKit is a great way to get acquainted with COSMOS operating system and its basic functionalities and modules.

3. PROCEDURE FOR DEPLOYING COSMOS OPERATING SYSTEM

Due to the integration of the COSMOS operating system in MS Visual Studio, it is necessary that this development environment be installed. According to the official COSMOS Website, it is necessary to use MS Visual Studio 2017 together with .NET Core Tools and .NET Framework 4.6.2 Developer Pack [12]. This means that older versions of development tools do not support working with COSMOS Operating System.

The program code of COSMOS operating system, either the version downloaded from the official Website or the updated version, is translated using the .NET translator. During the translation, the source code is being translated into the Common Intermediate Language (CIL), native language of the .NET Framework. For translating .NET CIL into machine language, a translator of IL2CPU (Intermediate Language to CPU) was developed within the COSMOS project [13].

Within the Cosmos Builder application, the designer chooses an option that determines how the project will finally be made. Among other things, these options define the way the project is launched. The launching features are Quick Emulator (QEMU), Virtual PC, and VMWare.

After installation of the UserKit, Visual Studio 2017 is updated with Cosmos section. From there, the user can choose one of the four available options for creating COSMOS operating system in C#, F#, or VB.NET programming language (Fig. 2).

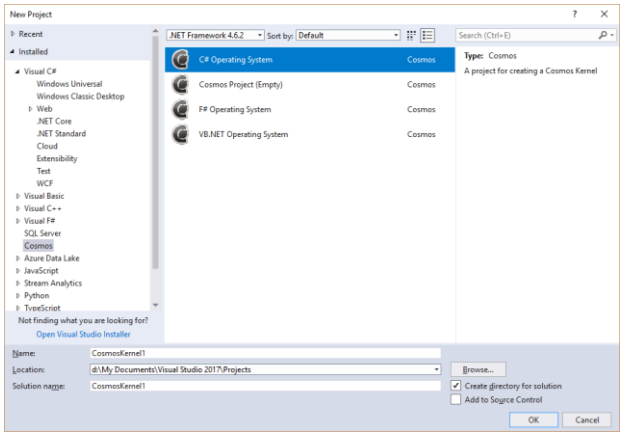

Figure 2. *Visual Studio options for creation of new COSMOS operating system*

We used C# as our programming language of choice. After naming the project, the user is presented with basic operating system with the functionality of getting input from the user and echoing it back.

The Fig. 3 shows the composition of the COSMOS OS Visual Studio solution. It is consisted of two projects, C# project with **Kernel.cs** main source file, and **CosmosBoot** project, referencing the

underlying core functionalities and user's C# project.

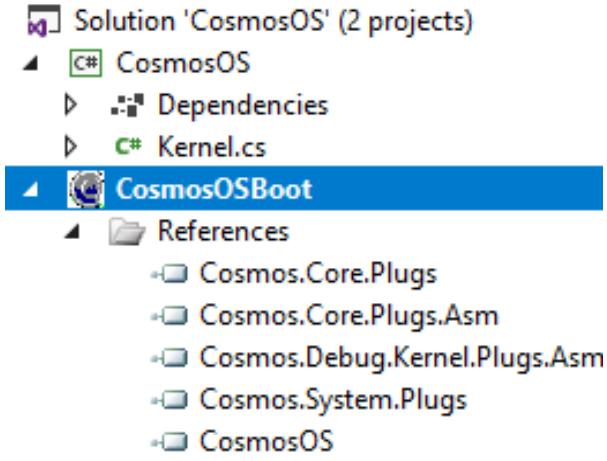

Figure 3. *COSMOS OS Visual Studio solution constitution*

The main class is called **Kernel** and it inherits the **Sys.Kernel** class of **Cosmos.System** namespace. Two methods have to be overridden: **BeforeRun()**, that executes before booting up the operating system and **Run()**, that actually holds the key part of the operating system.

Deployment options for COSMOS operating system are given in Fig. 4. These are Hyper-V, Intel Edison Serial boot, ISO image, PXE Network Boot, USB Bootable Drive, and VMware. In our case, we used VMware, with option to deploy to VMware Player or Workstation edition. This is the easiest way for ongoing development and debugging.

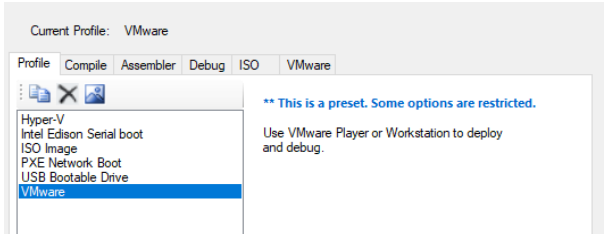

Figure 4. *Deployment profiles for COSMOS operating system*

The solution is executed in debug mode and since VMware is our deployment option, a new virtual machine is created and COSMOS is booted (Fig. 5).

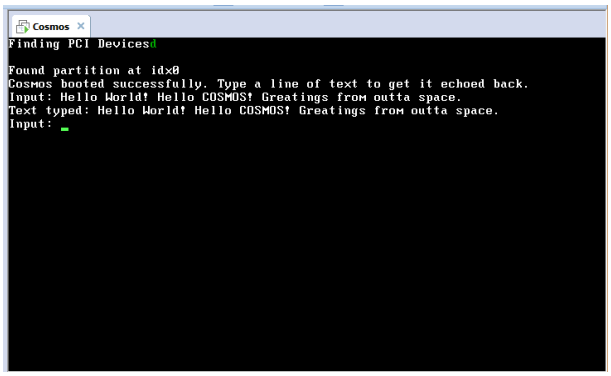

Figure 5. *Output from simple COSMOS operating system*

Default virtual machine options are as follows:

- Memory size: 256 MB
- Number of processors: 1
- Hard disk size: 512 MB
- CD/DVD: Using file ./CosmosOSBoot.iso.

Going beyond basic functionalities of COSMOS operating system is relatively difficult, since it requires writing separate modules for handling access to different rings. At the User ring, the program code can access only areas of User and System ring. Furthermore, usage of .NET methods is restricted to only few namespaces, and beyond that, the plugs have to be made.

We realized a simple command-line operating system with only four commands:

- **help**: shows the available commands
- **about**: prints the operating system UserKit version
- **reboot**: reboots the machine
- **miv**: enters MIV file editor [14].

In the **BeforeRun()** method, a file system is initialized and user is asked to enter his username. From now on, the system displays **{username}@CosmosOS >>** command prompt, where **{username}** is replaced with actual username provided, as shown in Fig. 6.

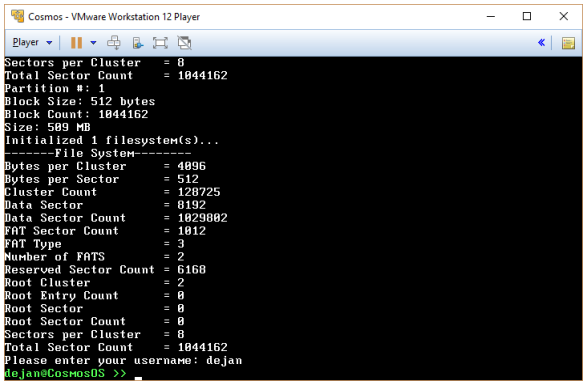

Figure 6. *Welcome screen of COSMOS OS*

An example of available commands is shown in Fig. 7.

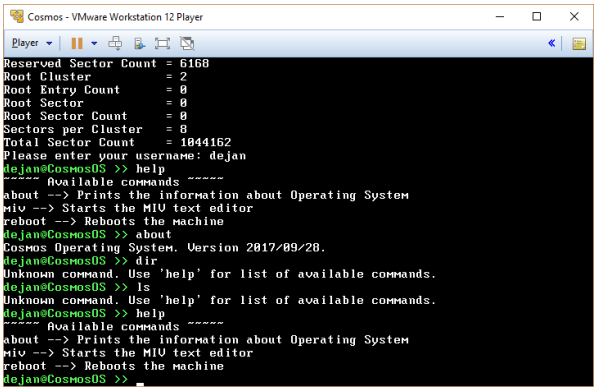

Figure 7. *An example of available commands*

The most interesting part about realized operating system is its ability to use file system, create,

edit, and display content of text files. Usage of MIV text editor is shown in Fig. 8.

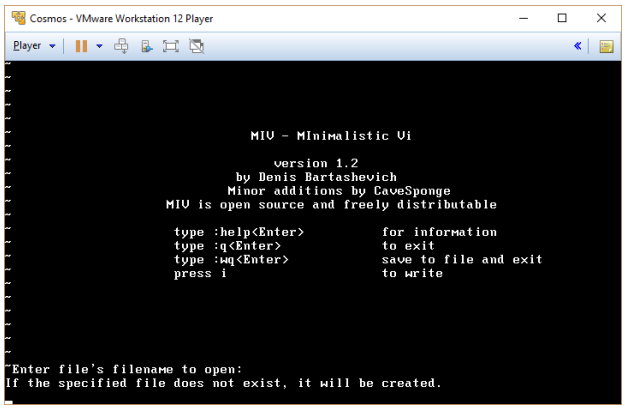

Figure 8. *Welcome screen of MIV text editor*

MIV text editor is accepting the following commands:

- **help**: display information
- **q**: exit text editor
- **wq**: save content to file and exit text editor
- **i**: write to file.

4. CONCLUSION

Operating systems have passed a long way from batch processing, single-user, single-task command-line interface to multi-user, and multi-task graphical user interface with service-oriented realization. The knowledge of their principles of operation and realization is one of the fundamental part of the Computer Science students' education.

Modern operating systems that are most commonly used are either non open-source or too complex for research and teaching the principles of operating systems. Thus, the educational operating systems provide students with relatively easy way of getting to know their implementation and fundamental operation modes.

COSMOS operating system, presented in this paper, belongs to such group of educational operating system. It is complex enough to provide different access privileges through its system of rings and plugs, but on the other hand, relatively easy to implement some basic ideas and principles. In this paper, we developed a simple command-line operating system with few commands and file system functionality. Its connection with C# programming language and Visual Studio development environment facilitates its realization among students and other hobbyists that are used to Visual Studio development.

It should be noted that there are several successful implementations of COSMOS operating systems [15], some of which are with graphical user interface and large number of functionalities that can be found in modern day operating systems.

ACKNOWLEDGEMENTS

Research presented in this paper is supported by the Ministry of Education, Science, and Technological Development of the Republic of Serbia through national project no. TR32043, for 2011 – 2018 period.

REFERENCES

[1] Clemons, L. C. (2012). *Understanding Computer Operating Systems: Apple Macs's, Linux, Unix, Windows*. CreateSpace Independent Publishing Platform

[2] Meike, G. B. (2018). *Inside the Android OS: Building, Customizing, Managing and Operating Android System Services*. 1st Edition, Addison – Wesley Professional

[3] Renner, T. (2011). Mobile OS – Features, Concepts and Challenges for Enterprise Environments. *International Journal of Advancements in Research & Technology*, *2*(3)

[4] Divya, K., KrishnaKumar, S. V. (2016). Comparative Analysis of Smart Phone Operating Systems Android, Apple iOS and Windows. *International Journal of Scientific and Applied Science (IJSEAS)*, *2*(2), 432 – 438

[5] Galakhov, D. (2017). *Creation of Educational Operating System: Using the C# and .NET Framework*. Bachelor's thesis, South-Eastern Finland University of Applied Sciences

[6] Thoma, J. (2016). *Linux Systems for the Educations Sector*. Linux Magazine

[7] Tanenbaum, A. S., Woodhull, A. S. (2006). *Operating Systems: Design and Implementation*. 3rd Edition, Prentice Hall

[8] Kudzu, C. Z. H. (2010). *Develop your own Operating System in C# or VB. NET*. Code Project Articles, available at: https://www.codeproject.com/Articles/99928/Develop-Your-Own-Operating-System-in-C-or-VB-NET

[9] *Preboot Execution Environment Specification*. Version 2.1, Intel Corporation/SystemSoft, 1999

[10] *CosmosOS Wiki*, available at: https://github.com/CosmosOS/Cosmos/wiki

[11] Syslinux Repository, available at: http://repo.or.cz/syslinux.git

[12] Nagel, C. (2018). *Professional C# 7 and .NET Core 2.0.* 7th Edition, Wrox

[13] Daumler, M. (2015). *Real – time Code Generation in Virtualizing Runtime Environments*. Doctor of Engineering Dissertation, Chemnitz University of Technology, Department of Computer Science

[14] *MIV – CosmosOS Text Editor*, available at: https://github.com/bartashevich/MIV

[15] *CosmosOS Projects*, available at: https://github.com/CosmosOS/Cosmos/wiki/Cosmos-Projects