From Geiger-Counters to File Systems: Remote Hardware Access for the Operating Systems Course

http://dx.doi.org/10.3991/ijoe.v12i09.6102

J. Wolfer, and W. J. Keeler
Indiana University South Bend, South Bend, USA

***Abstract*—Operating systems interface between hardware and the user, random numbers are useful for security and simulation, and file systems form the program access to them in a modern operating system. Blending these items into a remotely accessed infrastructure forms the basis for supporting operating systems projects. This work describes the hardware, software, and communication infrastructure to support student projects by sharing remote hardware to acquire background radiations events with a Geiger counter, transforming those events into random numbers, and providing those numbers through a custom file system. Collectively, the hardware and software provide an inexpensive remote laboratory experience for computing students.**

***Index Terms*—operating systems; random numbers; file systems; pedagogy; remote laboratory; Geiger Counter; Raspberry Pi**

I. INTRODUCTION

Remote access to scarce or potentially dangerous laboratory experiences is an emerging trend in engineering education. This has the effect of multiplying the equipment investment, and in the case of education, providing laboratory access to a larger contingent of students. This, in turn, can be crucial when attempting to bootstrap critical engineering education infrastructure with limited resources [1,2]. In addition to the obvious advantage of enhanced investment, remote resource access also brings a level of convenience to the user, laboratories may be available at a place and time of the student's choosing, for example.

While remote engineering laboratories are emerging due, in part, to the rich internet communication infrastructure, the remote model was adopted in the earliest days of modern computing. Exploiting the time-sharing paradigm at the operating system level many computers allowed "modem-access" to share resource. The UNIX time-sharing system may be the most recognizable [3] example of such a system. That having been said, it is not as common for computer science students to have shared, low-level, access to hardware components as is becoming prevalent in other areas of engineering. Since an operating system must interface with hardware we wished to provide a moderately low-level hardware experience for our students in the Operating Systems class. To that end we developed a shared laboratory experience using a consumer-grade Geiger counter to form the basis for students to develop software to synthesize random numbers from background radiation events and to make those numbers available by developing a custom file system in user space.

The balance of this report will provide a background for remote laboratories, describe the operating systems class, the Geiger counter hardware, random numbers acquired, and the file system project.

II. BACKGROUND

When designing remote laboratory experiences it is important to consider local objectives in the context of those articulated in the broader intellectual community. For example, Restivo, et. al, summarize a variety of important objectives in the context of tactile experiences in virtual laboratories. These objectives include learner-centered items such as meeting the needs for diverse learning styles, provide experimental results to the students for subsequent analysis, to correlate laboratory results with theoretical concepts, and to provide a basis for lifelong learning [4].

Two additional, important, considerations emerge when considering design and deployment of both the remote laboratory and the student projects designed around it. Those are the combination of so-called "soft skills" and the elusive student motivation. While not directly addressing pedagogy, Edwards, Tovar, and Soto make the observation that employers expect students to have skills beyond the technical, including teamwork, leadership, and the ability to communicate in written and oral form [5].

In a similar vein, Settle and Sedlak report the results of a survey designed to ascertain faculty attitudes toward computing student motivation quantitatively. Their analysis indicated that the majority of computing educators perceive motivation "to be important in all learning situations, but particularly…in courses with significant obstacles, and in particular areas of computing, including theoretical courses [6].

To meet these objectives engineers and educators have pursued a variety of approaches including team projects, flipped classrooms, virtual and remote laboratories, and other implementations of active learning. For example, Mason, Shurman, and Cook did a careful comparison of a senior engineering course in traditional and inverted, or flipped, formats. They report that while it took students up to four weeks to adapt to the inverted classroom, the ultimate outcomes were similar or better than that for the traditional classroom [7].

Likewise, Kyle, et al., show the design of a project-based bioinstrumentation course [8]. Students were to design, test, and implement biomedical signal measurement apparatus. They report that this approach generates "highly exciting instructional platforms for both teachers and students."

There are examples in the computing domain as well [9-12]. For example, Arbelaitz, Martin, and Muguerza report good results for an active learning approach to computer architecture [12]. Using a problem-based learning format requiring approximately sixty hours to complete, they reported that student learning was positively impacted. They also observe that "the students prefer to learn through carrying out a real project because they feel it provides them with more motivation to learn." Similarly, Uhsadel, et al., use teams of two to develop a public-key cryptography application using a combination of hardware and software. The students define the tradeoff between the hardware and software components, thus providing a platform for using both hard- and soft-skills in their implementation [13].

Consistent with these efforts to enhance motivation and learning, we approached the Operating Systems class at our institution with a series of sustained mini-projects, culminating in the system described in this report. The hardware and software infrastructure described here was first demonstrated at the REV 2016 conference, and was developed to support the final project as an active-learning, team oriented, exercise for the Computer Science Operating Systems class [14].

III. OPERATING SYSTEMS

Like others [15], the Operating Systems course as taught at our institution combines the study of operational principles with significant hands-on implementation. It is considered a capstone course, and is offered near the end of their undergraduate program, typically the end of their senior year. Classroom topics include both operating systems principles and hands-on implementation. Topics include general system architecture, interrupt and clock structures, system calls, inter-process communication, process scheduling and management, memory organization and management, file systems, device interfaces and drivers, and contemporary security issues.

In addition to the topical lectures described above, the course includes a series of mini-projects to be developed and implemented. Examples include probing the operating system kernel for debugging information, modifying the process scheduling algorithms, and developing file systems and/or device drivers. These are accomplished using a combination of operating systems such as MINIX and Linux.

As a capstone class it is assumed that, in addition to the technical content, the class will consolidate their analytical abilities with appropriate soft-skills. To facilitate the soft-skill integration into the final project students are divided into teams of four or five. Each team is self-organizing, dividing the work assignments as they see fit, and reporting progress weekly to the instructor. In addition, some class time is allocated to team meetings where the instructor can observe and interact with the team to both assess progress and advise where necessary.

IV. PROJECT DESCRIPTION

One important aspect of this class is giving students experience with both device interfaces and file system level access to device information. Typically a file system provides programs, through the common read and write system calls for example, access to bytes of data stored on an external medium. For example, reading from a disk drive. That concept is abstracted in UNIX-based systems to include reading from devices acting as files. For example, reading from stdin, the standard input device (keyboard), or writing to /dev/null as a data sink. For this project students develop a system that, when read, provides random numbers synthesized from Geiger counter events. This requires them to interact with two ends of the acquisition subsystem; background data acquisition (synthesizing random numbers from Geiger counter events) and the file system interface.

The choice of physically-based random number generation is based on the concept that real-life projects are motivating for most students. Random numbers are vital for both security and simulation. Our modern cryptographic systems depend on them. That having been said, many random number libraries provide algorithmically produced, pseudorandom, numbers. These can be sensitive to initial condition, and even risky in situations requiring security [16]. Likewise, many simulations rely on random numbers and can be affected by the quality of those numbers [17].

There are a variety of physically-based random number generators available, ranging from electrical and thermal noise to radiation events. A survey of these, as well as pseudo-random number generators, can be found in [18]. We chose to use a Geiger counter since it is visual (an LED glows for each event) so students could visually correlate what was happening in their programs with the hardware output.

V. HARDWARE

A. Cluster Overview

Once the students acquire Geiger counter event timestamps and synthesize random numbers from them, they then develop a file system in user space (FUSE) to allow program access to those numbers

The hardware supporting this effort has been developed across two generations of Raspberry Pi computers. Initially we augmented a previously designed cluster developed to illustrate supercomputing principles [19-21] with a Geiger counter interface, networking software, and additional compute capability. This cluster is shown in Fig. 1. Designed primarily to support pedagogy in the high-performance computing class, the cluster consisted of four Raspberry Pi computers [22] and an Nvidia Jetson TK1 [23] which includes a quad-core cpu coupled to 192 core CUDA card. The TK1 was not deployed for this project, but the cluster was augmented with two additional Raspberry Pi computers, for a total of six units. That configuration supported the initial deployment for the Operating Systems class, supporting a total of thirty-one students.

Since this system was originally deployed, upgraded Raspberry Pi computers have become available. Consequently, the system has been reduced in size, but upgraded with new capabilities as described below. The originally

deployed system with Geiger counter interface is shown in Fig. 1, the enhanced system is shown in Fig. 2.

B. Raspberry Pi

The Raspberry Pi (Fig. 3) has become a popular, credit-card, sized computer system capable of supporting a variety of operating systems. The original model used for this project was the "B+" model, featuring:

- Broadcom BCM2835 ARM CPU, 700 Mhz.
- 512 MB RAM
- 10/100 Mbps Ethernet
- 16 MB micro SD card acting as a disk drive
- 4 USB 2 ports, audio in/out, composite video, and HDMI outputs.

In addition, the Raspberry Pi shares capabilities found in other embedded systems. Specifically, the modules expose a variety of General Purpose IO (GPIO) pins enabling interfacing to the physical world, including to the Geiger counter used here.

The upgraded Raspberry Pi, while identical in form-factor, quadruples the number of enhanced CPU's and increases the baseline clock speed and memory. Specifically, each model 2 includes

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM

C. Geiger Counter and Interface

The Geiger Counter module forming the basis for random numbers in this project is the Sparkfun SEN-11345 [24] is shown in Fig. 4. While this unit is capable of providing data through a USB port, we elected to provide an interface directly from the buffered output from the GM tube to GPIO inputs on the Raspberry Pis. This supports the students by providing an environment for low-level interface handling. Note that the Raspberry Pis use 3.3v logic while the Geiger counter is 5v, so appropriate level translation is required. The output of the Geiger counter was buffered through Schmitt triggers and connected to appropriate GPIO pins of each Raspberry Pi, giving each system the same input data.

VI. NETWORKING SUPPORT

One critical component for system deployment is to be able to integrate into the university network infrastructure. This involves meeting security requirements and exporting student accounts and file systems to the Raspberry Pi computers. This allows students to remotely login to the Pis in the same manner, and having the same available resources, as any of their accounts on a "larger" computer.

To accomplish this the Raspberry Pis had the Raspbian Linux distribution installed, patched, and updated with security and kernel packages. The operating system was then configured to work with the departmental NFS server which provides student files. Login credentials are authenticated through a combination of LDAP and Kerberos configured to require initial login into an existing university computer followed by a second login to a specific Raspberry Pi identified by it's IP address. This two-level security allowed for easy remote access with enhanced security.

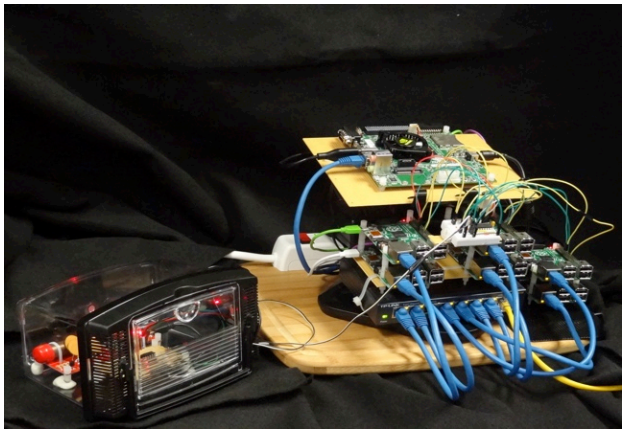

Figure 1. Original Cluster

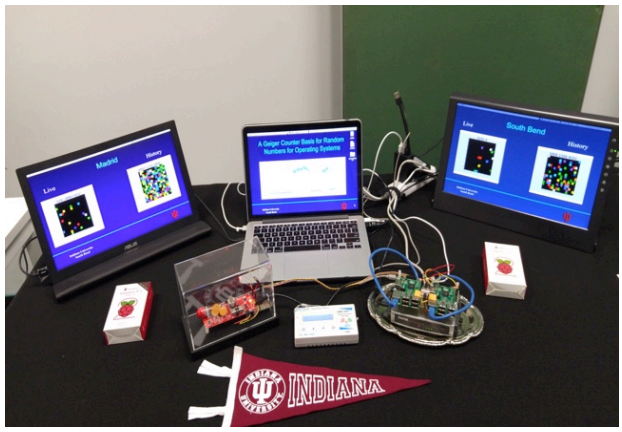

Figure 2. Enhanced Cluster

Figure 3. Raspberry Pi

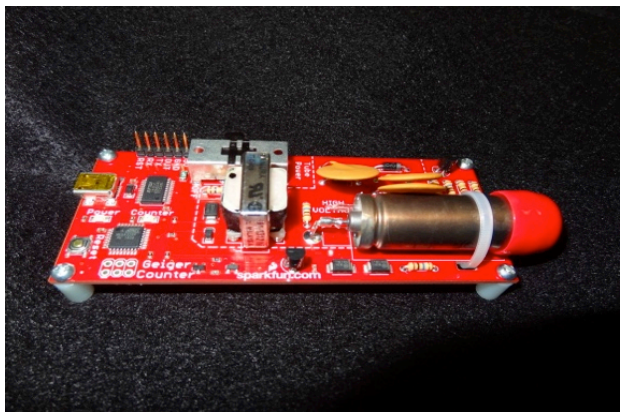

Figure 4. SparkFun Geiger Counter

VII. RANDOM NUMBER ACQUISITION

As indicated earlier, random numbers are important for security, encryption, and simulation. Software-only approaches attempt to seed a generator from entropy in the system, such as time, and form an algorithmic series of pseudo-random numbers. Certain physical phenomena, such as radiation events and thermal noise, are believed to be truly random [25]. Comparing the time difference between successive pairs of radiation events generates a single bit. These timestamps are harvested to create random bits, bytes, and words as necessary.

To support the acquisition of the timestamps, along with other aspects of this project, a variety of open-source software was deployed. This includes the GNU compiler suite (C, C++), Python, including SciPy, Numpy. This was supplemented by the pigpio [26] library to help manage GPIO access to the Geiger counter, and the FUSE and Python FUSE libraries to support file system development.

The pigpio library formed an essential part of this work. An open-source software, it allows for managing signals on the general-purpose IO pins of the Raspberry Pi computers and makes these signals available to multiple users simultaneously through a variety of software interfaces, including a callback API.

Fig. 6 shows sample code for Geiger counter event collection. When the Geiger counter indicates an event on GPIO pin 23 of the Raspberry Pi the pigpio library intercepts the signal on the falling-edge, which triggers a callback to the function registered, in this case "mycb". The "mycb" function, in this case, gets the timestamp for the event from the operating system and saves that timestamp to a file for future use. It should be noted that there is some time ambiguity in this approach since these are multi-user, multi-process, operating systems so there could be some bias introduced for the random numbers.

A visual inspection can be used as a quick check of the plausibility of the random numbers being generated [27] as shown in Fig. 5. In this image each bit in a number is mapped to a single pixel which is set to either dark or light corresponding to the bit value. Notice the lack of obviously apparent pattern in the image.

While the visual test is a reasonable first look at the random numbers, there are other available test suites. One commonly used test suite in the public domain is the ENT pseudorandom number sequence test program [28]. This program implements a variety of tests such as measuring the entropy, compressibility, mean, Chi square, Monte Carlo estimate for Pi, and a serial correlation coefficient. Table 1, under Pre-Bias Removal, shows the results of running this program on 1.2 million timestamps. Notice that, while very good, there is some residual bias in the sequence. Table 1, Post-Bias Removal, shows the data after removing the bias [29].

Finally, Fig. 7 shows an application using the bits to generate an image live on the web [30]. In this image twelve random bits are used to form three 4-bit random numbers, the first of which selects one of sixteen rows, the second selects one of sixteen columns, and the third designates one of sixteen colors to display at location image[row,column].

Students are given wide latitude to design and create the specific mechanism for using/creating the random numbers in their individual systems. For example, since

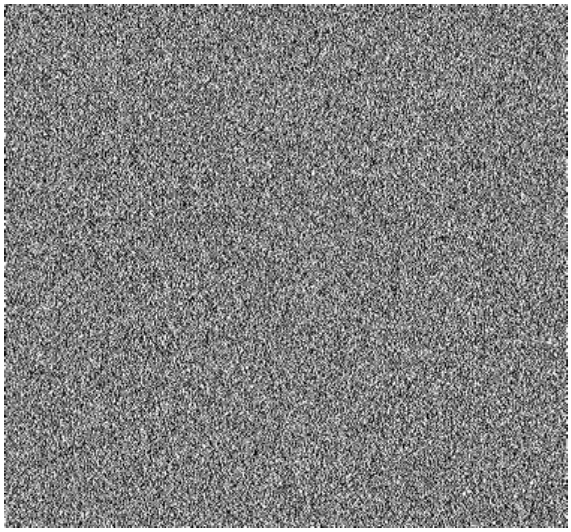

Figure 5. Random Bit Image

```
#!/usr/bin/python
import time,pigpio

fo=open("randtimegeiger.txt","a")
def mycb(x,y,z):
        t=time.time()
        print t
        fo.write(str(t)+'\n')
        fo.flush()

pin=pigpio.pi()
cb=pin.callback(23,pigpio.FALLING_EDGE,mycb)
```

Figure 6. Pigpio Program Fragment

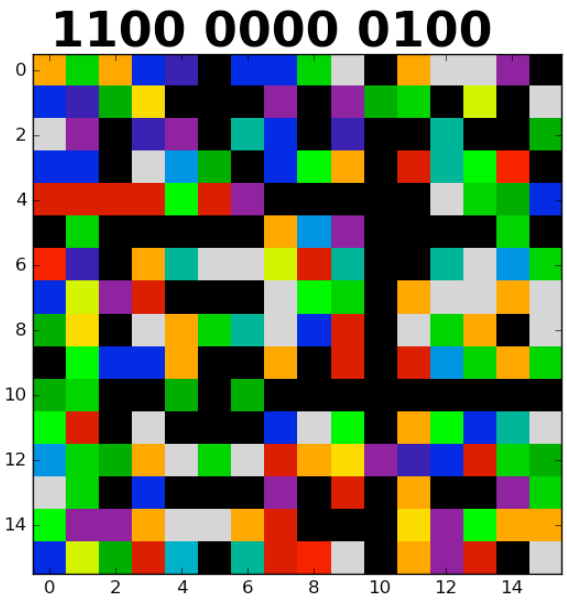

Figure 7. Snapshot Live Random Number Web Demonstration

TABLE I. ENT RANDOM TEST

	Pre-Bias Removal	Post-Bias Removal
Entropy	1	1
Chi-square	0.671601	0.328496
Mean	0.499635	0.50008
Monte-Carlo-Pi	3.458389	3.137899
Serial-Correlation	-0.333256	-0.000454

background radiation provides relatively infrequent events, on the order of thirteen to twenty counts/minute at our location, it is not unusual for a data request to exhaust the supply of truly random bits. At that point the student can, for example, use the last remaining truly random bits to seed a conventional pseudo-random number generator until sufficient random data is reacquired. Others would continually acquire bits in the background and save the data over days and weeks, creating a larger pool of numbers.

VIII. FUSE FILE SYSTEM

Once random numbers are acquired, students are expected to develop their own File system in User Space (FUSE) to provide user access to the random numbers. The use of FUSE files systems within an operating systems class has been discussed in [31]. As a secondary function, the student file system could also record the background radiation rate into a log file.

FUSE [32] is a cooperative, call-back based interface to the Linux file system infrastructure. The FUSE layer has been used to implement a variety of useful file systems ranging from NTFS [33] to encrypted file systems [34]. In operation the user requests service through the normal system call interface, trapping to the kernel. The Linux Virtual File system layer validates the request and activates the FUSE specific API, which then relays the validated request to the user file system via callbacks defined in the API. Since the actual reading and writing is implemented in user space, all of the normal development tools are available and API bindings to a variety of computer languages can be easily provided. Thus students can work in a language familiar to them, or can exploit development efficiencies in high level scripting languages such as Python.

While the development occurs in user-space, and the kernel does a great deal of the detailed communication and validation work, the user is still responsible for implementing all the traditional aspects of a typical file system including directory structure, path management, file creation, deletion, modification, as well as any associated metadata.

Given the FUSE libraries and Geiger Counter access, the students proceed with the task of creating a file system that a) provides random numbers to the user program when the file is read and b) logs the current estimate of counts per minute when a file within the files system is written to.

IX. DEPLOYMENT AND FUTURE WORK

Between the deployment of the original cluster and the enhanced cluster this system has served approximately sixty students divided into ten implementation teams. Although the system is capable of serving the Geiger counter data directly over the network, the file systems were developed remotely on the Raspberry Pi. The system was deployed and ran continuously, 24/7, for a total of over sixty consecutive days without failure.

Building on the successful deployment, we would like to extend this work in at least two directions. The first is to do a systematic assessment of student attitudes and outcomes now that we have a demonstrated, stable platform to work with. The other is to integrate additional hardware-based approaches to random number generation into the system for comparison.

X. CONCLUSION

We show the development, implementation, and deployment of an end-to-end, hardware and software, remote platform supporting the development of file systems for the computer science Operating Systems class. We believe that this provides a rich experimental environment for student projects.

REFERENCES

[1] A. Naddami, A. Fahli, M. Gourmaj, A. Pester, and R. Oros, "Importance of a Network of Online Labs in Magrebian Countries", *REV 2014 International Conference on Remote Engineering and Virtual Instrumentation*, pp. 77-78, 2014.

[2] R. Salah, G. R. Alves, and P. Guerreiro, "Reshaping Higher Education Systems in the MENA Region: The Contribution of Remote and Virtual Labs", *International Conference on Remote Engineering and Virtual Instrumentation (REV)*, pp. 240-245, 2014. http://dx.doi.org/10.1109/rev.2014.6784265

[3] D. Ritchie and K. Thompson, "The UNIX Time-Sharing System", *Communications of the ACM, vol. 17 number 7*, pp. 365-375, July, 1974. http://dx.doi.org/10.1145/361011.361061

[4] M. T. Restivo, A. M. Lopes, L. D. Machado, and R. M. Moraes, "Adding Tactile Information to Remote & Virtual Labs", *IEEE Global Engineering Education Conference*, 2011. http://dx.doi.org/10.1109/educon.2011.5773287

[5] M. Edwards, E. Tovar, and O. Soto, "Embedding a Core Competence Curriculum in Computing Engineering", *ASEE/IEEE Frontiers in Education Conference*, 2008.

[6] A. Settle and B. Sedlak, "Computing Educator Attitudes about Motivation", *unpublished Technical Report:* arXiv:1603.08996v1, 2016.

[7] G. Mason, T. R. Shuman, and K. E. Cook, "Comparing the Effectiveness of an Inverted Classroom to a Traditional Classroom in an Upper-Division Engineering Course", *IEEE Transactions on Education*, vol. 56 number 4, pp. 430-435, November, 2013. http://dx.doi.org/10.1109/TE.2013.2249066

[8] A. M. Kyle, D. C. Jangraw, M. Bouchard, and M. E. Downs, "Bioinstrumentation: A Project-Based Engineering Course", IEEE Transactions on Education, vol. 59, number 1, pp. 52-58, February, 2016.

[9] V. Cardenas, L. Azucena, F. Bertacchini, L. Gabriele, A. Tavernise, D. Elizabeth, O. Vizueta, P. Pantano, and E. Bilotta, "Surfing Virtual Environment in the Galapagos Islands", *International Conference on Remote Engineering and Virtual Instrumentation (REV), pp. 192-198*, February, 2015.

[10] K. Dickmann and A. A. Kist, "Remote Network Laboratory Development", *International Conference on Remote Engineering and Virtual Instrumentation (REV), pp. 370-374*, February, 2014.

[11] P. Abreu, M. R. Barbosa, A. M. Lopes, "Virtual Experiment for Teaching Robot Programming", *International Conference on Remote Engineering and Virtual Instrumentation (REV), pp. 395-396*, February, 2014. http://dx.doi.org/10.1109/rev.2014.6784199

[12] O. Arbelaitz, J. I. Martin, and J. Muguerza, "Analysis of Introducing Active Learning Methodolgies in a Basic Computer Architecture Course", *IEEE Transactions on Education*, vol. 58, no. 2, pp. 110-116, May, 2015. http://dx.doi.org/10.1109/TE.2014.2332448

[13] L. Uhsadel, M. Ullrich, A. Das, D. Karaklajic, J. Balasch, I. Verbauwhede, and W. Dehaene, "Teaching HW/SW Co-Design with a Public Key Cryptography Application", *IEEE Transactions on Education*, vol. 56, no. 4, pp. 478-483, November, 2013. http://dx.doi.org/10.1109/TE.2013.2257785

[14] W. J. Keeler and J. Wolfer, "A Raspberry Pi Cluster and Geiger Counter Supporting Random Number Acquisition in the CS Operating Systems Class", *International Conference on Remote Engineering and Virtual Instrumentation (REV)*, pp. 344-345, February, 2016.

[15] Nieh, J. and Vaill, C., "Experiences Teaching Operating Systems Using Virtual Platforms and Linux", *ACM SIGCSE 2005*, 2005.

[16] Z. Gutterman, B. Pinkas, and T. Reinman, "Analysis of the Linux Random Number Generator", *IEEE Symposium on Security and Privacy,* 2006. http://dx.doi.org/10.1109/sp.2006.5

[17] C. J. A. Bastos-Filho, J. D. Andrade, M. R. S. Pita, and A. D. Ramos, "Impact of the Quality of Random Numbers Generators on the Performance of Particle Swarm Optimization", *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4988-4993, October, 2009. http://dx.doi.org/10.1109/icsmc.2009.5346366

[18] A. A. Thomas and V. Paul, "Random Number Generation Methods a Survey", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6, no. 1, pp. 556-559, January, 2016.

[19] J. Wolfer, "A Heterogeneous Supercomputer Model for High-Performance Parallel Computing Pedagogy," IEEE Global Engineering Education Conference-ITEP, March, 2015. http://dx.doi.org/10.1109/educon.2015.7096063

[20] J. Wolfer, "A Model Supercomputer for Instructional Support (demonstration)," exp.at'15, The Third International Conference for Online Experimentation, June, 2015.

[21] J. Adams, J. Castwell, S. J. Matthews, C. Peck, E. Shoop, D. Toth, and J. Wolfer, "The Micro-Cluster Showcase: 7 Inexpensive Beowulf Clusters for Teaching PDC", Special Session: ACM Technical Symposium on Computing Science Education, March, 2016. http://dx.doi.org/10.1145/2839509.2844670

[22] Raspberry Pi Foundataion, "Raspberry Pi", https://www.raspberrypi.org/

[23] Nvidia, "Jetson TK 1", http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

[24] Sparkfun, "Geiger Counter Random Number Tutorial", https://www.sparkfun.com/tutorials/132

[25] J. Keisey, B. Schneier, D. Wagner, and C. Hall, "Cryptanalytic Attacks on Pseudorandom Number Generators", Fast Software Encryption, Fifth International Workshop Proceedings, March, 1998.

[26] Pigpio, "pigpio library", http://abyz.co.uk/rpi/pigpio/

[27] D. Beznosko, T. Beremkulov, A. Duspayev, A. Iakovlev, A. Tailakov, and M. Yessenov, "Random Number Hardware Generator using Geiger-Mode Avalanche Photo Detector", unpublished preprint: arXiv:1501.05521, 2015.

[28] J. Walker, "ENT: A Pseudorandom Number Sequence Test Program", http://www.fourmilab.ch/random/, 2016.

[29] J. Mather, http://www.ciphergoth.org/crypto/unbiasing/, 2016.

[30] J. Wolfer, "Random Number Demonstration", http://www.cs.iusb.edu/~jwolfer/

[31] J. Wolfer, "Linux Experience in the General Operating Systems Class", International Conference on Engineering and Technology Education, March, 2014. http://dx.doi.org/10.14684/intertech.13.2014.42-44

[32] FUSE, "File system in User Space", https://github.com/libfuse/libfuse

[33] Ntfs3g,"NTFS FUSE File system", http://www.tuxera.com/community/open-source-ntfs-3g/#tab-1414502495464-2-9

[34] V. Gough, "ENCFS Encrypted File System", https://github.com/vgough/encfs

AUTHORS

J. Wolfer is with the Department of Computer Science, Indiana University South Bend, South Bend, IN, 46634, USA.

W. J. Keeler, is with the Department of Computer Science, Indiana University South Bend, South Bend, IN, 46634, USA.

Submitted, 09 March 2015. Published as resubmitted by the authors on 09 April 2015.