

Stage 1: Operating System Features

This report will examine the different features of two popular open source UNIX-like operating systems:

1. Ubuntu 20.04 LTS (Focal Fossa)
2. FreeBSD 12.2

Both systems ultimately share many similarities, but there are differences that are important to compare when evaluating them for corporate use.

* 1. **OS Architecture, Scheduling and Memory Management**

When examining the architecture of an operating system, it makes sense to start at the kernel. The kernel is the core component of an operating system and is responsible for the most important day-to-day tasks of the operating system, such as interfacing with the hardware and scheduling processes. A potentially major difference between FreeBSD and Ubuntu is in their respective kernels. Ubuntu 20.04 uses the Linux kernel (alongside many other Unix-style operating systems such as Red Hat Linux and SUSE Linux), whereas FreeBSD uses its own free-standing kernel (McKusick, Neville-Neil and Watson. 2015). Two important kernel functions in any operating system are process and memory management. We will examine the implementations of these functions in FreeBSD and Ubuntu in turn.

Process management refers to the management of processes or programs running on a system. Both Ubuntu and FreeBSD use a ‘modes of operation’ model for processes, whereby a process either runs in *user mode* (with lesser privileges) or *kernel mode* (with higher privileges). FreeBSD’s default scheduler ‘ULE’ uses a time-slicing model (where each process is given a fixed amount of time for execution), with multiple queues. Processes themselves are split into different priorities, with real-time processes having a separate queue (and priority over) idle processes. Overall, FreeBSD’s scheduling model favours interactive programs (McKusick, Neville-Neil and Watson. 2015). Ubuntu uses the Linux kernel’s ‘Completely Fair Scheduler’ (CFS). CFS also uses time-slicing and ‘weights’/priorities (Lozi, Jean-Pierre et al. 2019). However instead of traditional queues, the CFS uses a type of binary tree called a *red-black tree* whereby each task’s ‘virtual runtime’ is stored. The left child in the tree always has the lesser runtime, and therefore CFS is able to use the tree to schedule processes via their runtime. In a detailed performance analysis of both schedulers, Bouron et al. (2018) found that there was no clear winner. It was noted that although FreeBSD’s *ULE* scheduler achieved better process load balancing over a long period of time, the Linux kernel’s CFS handles changes of workload (e.g. from small to high) better. Therefore, Ubuntu may be the superior choice in systems with an erratic workload, whereas FreeBSD may favour stable, heavy workloads. It is noted that like the ULE scheduler, CFS favours interactive programs (Katlin, M. 2019). This makes both schedulers suitable for modern desktop applications.

As memory comes with a high price tag in corporate environments, the memory management of these two operating systems is another important aspect to consider. Unsurprisingly, both operating systems (as Unix-like systems) have similar memory management techniques. For example, both operating systems use a ‘virtual memory subsystem’ (Dube, R. 1998) that incorporates paging and swapping. Slight divergences do exist, for instance Dube in his comparison of both memory management subsystems highlights differences in swapping out idle processes. Ultimately, both operating systems are trusted for use in high performance environments. FreeBSD is used in Netflix’s content delivery network that accounts for 15% of all downstream internet traffic (Looney, J. 2019.), whilst Ubuntu is the most used guest OS in both AWS and Azure (Canonical). Therefore, it is not expected that there will be considerable differences in memory performance and analysis between Ubuntu’s Linux kernel and FreeBSD’s kernel.

Fundamentally despite some differences, the two kernels have much in common. Both kernels are ‘monolithic’ kernels (McKusick, Neville-Neil and Watson. 2015), meaning that the operating system is responsible for all of the base requirements of the system, and these functions reside in privileged address space. Furthermore, a Linux compatibility package exists for FreeBSD which enables Linux programs to run in FreeBSD (FreeBSD Handbook, 2020). This means that in a corporate environment, users could be offered either operating system with the same Linux applications used by both. In fact, the some of the main distinctions between the two operating systems lie in design philosophy and notional, not architectural, differences. FreeBSD is a complete software package that includes a kernel and user space applications packaged into a general ‘operating system’ (Greg Lehey, 2018). Alternatively, Ubuntu uses in the Linux kernel and provides its own distribution specific applications on top of this kernel. The supremacy of either of these two designs is subjective. On the one hand, the centralization of the FreeBSD model means that the FreeBSD community can focus on offering a consolidated operating system package. Lehey comments in his *‘FreeBSD Quickstart Guide for Linux Users’* that the “all from one supplier” nature of FreeBSD means that system upgrades are easier. On the other hand, Ubuntu can draw from a larger pool of contributors that add to the general development of the Linux kernel. The popularity and extensive use of Linux may also allow Ubuntu to support a larger array of applications and devices than FreeBSD.

* 1. **File System**

Both FreeBSD and Ubuntu use a software abstraction layer known as the ‘virtual file system’ or VFS (Yang, K. and Wallace, K. 2011). The VFS presents a single interface to processes on a system that allows standard operations (e.g. open, close, read, delete) to be performed on files without interaction with the underlying physical file system. This means that both systems support a wide variety of file systems. Subsequently in a corporate environment it is unlikely that IT personnel would need to concern themselves with file system compatibility, as both FreeBSD and Ubuntu can use NTFS to interact with the Windows file system, or indeed rely on the VFS to read each other’s file systems. When examining the difference between file systems, it perhaps makes more sense to only focus on the default or *native* file systems for each OS. FreeBSD has two native file systems, the Unix File System (UFS) and the Z File System (ZFS), whereas ext4 is the default file system for Ubuntu. ZFS is the newer of both FreeBSD file systems, and has many features that make it attractive for use in a corporate environment or in production servers (like web servers or file servers). These features include the ability to act as a volume manager on top of a file system, for example ZFS provides its own software RAID known as *RAID-Z* (Hickmann, B and Shook, K. 2007)*.* Although Ubuntu does support ZFS, this support is still in experimental mode only (Roche, D.). Therefore, to take advantage of the features offered by the “advanced file system” (FreeBSD Handbook, 19.) of ZFS, FreeBSD may be the more reliable choice.

Stage 2: Identified Problems & Solutions

There are two identified problems that must be resolved before the systems are approved for production. Each problem identified is resolved with the use of a shell script. Both scripts are written using the *bash* shell, with one script tested in Ubuntu and the other in FreeBSD. As both systems support the bash shell, each script is compatible with both.

1. **External backups (Ubuntu)**
   1. *Requirements*

We are aware that the file system in both operating systems takes steps to ensure the integrity of user files. However, neither file system actively takes steps to make external copies of such files out of the box. If end users are going to be accessing these operating systems and storing company data on them, then regular, reliable backups are critical. As user’s in Ubuntu Linux work within their ‘home’ directories, this is a good place to create user specific backups. Furthermore, there is an important security aspect to consider. Therefore, any backups taken should be encrypted.

*1.2. Test Plan*

Our test plan consists of the following:

1. Check that our solution can backup user files to an external directory, by running the script on test files.
2. Check that our files have arrived in the backup location without corruption.
3. Check that our files are secure and have been encrypted.
   1. *Script*



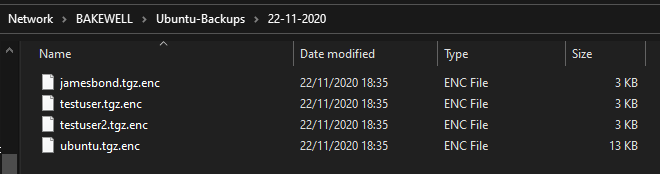
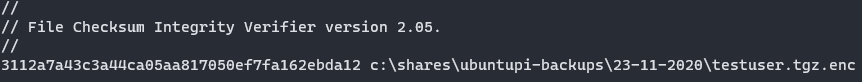
The above script creates a backup of each user directory in /home. The backups are encrypted and stored externally in a Windows file share. The script must be run using *sudo* permissions.



*1.4. Testing*

* The script was tested against four test user directories in /home:



* After running the script, we can confirm that backups of these directories have been created on our backup server. 
* We can check that our files have arrived without corruption by taking a hash of the file on the backup server and comparing it with a hash taken from the Ubuntu system.
* Hash of ‘testuser.tgz.enc’ taken from the backup destination using **SHA-1**:
* Hash of the same ‘testuser.tgz.enc’ file from the Ubuntu computer using **SHA-1**:
* As the hashes match, we know that both files are identical and have verified the integrity of our backup.
* Finally, we can use the simple ‘file’ command in Ubuntu to check that our files are encrypted:

1. **Monitoring Disk Usage (FreeBSD)**
   1. *Requirements*

Although both operating systems have tools to check the amount of disk space in use, this requires active intervention by the user. It would be more appropriate for a program to run on a schedule and alert the user or IT department if action was required. As the most important disk for the user is the disk that the root directory is mounted on, our solution will monitor the root directory only.

* 1. *Test Plan*

Our test plan consists of the following:

1. Check that our script produces an output providing the user with information on their current disk usage.
2. Check that this information is accurate, via comparison with other inbuilt tools.
3. Check that this script is set up to run on a schedule.

*2.3. Script*

Bash is not included in the default install of FreeBSD (Ferrell, J. 2020). Therefore, it is first required to install the bash package on the computer:

*sudo pkg install bash*

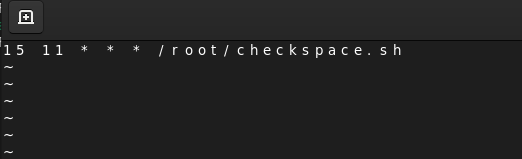
In addition, this script requires the ‘mailutils’ package to be installed. Unlike previously, this has not been written into the script and must be installed before running, like so:

*sudo pkg install mailutils*

The script is as follows:

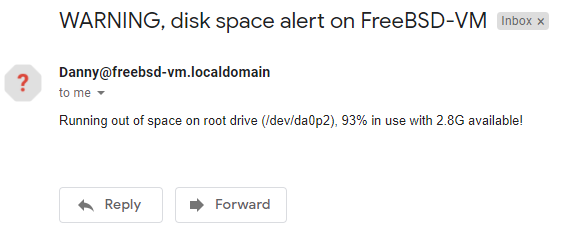


We want this script to run on a schedule, so that it can monitor disk usage at regular intervals. It is possible to use the *cron* program to do this. Our script has been set to run daily at 11:15am using the below setting in the root user’s crontab file:

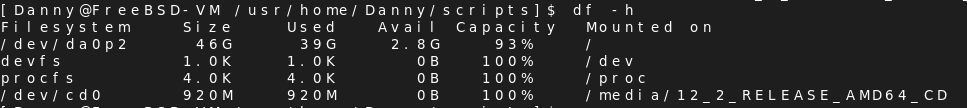


*2.4. Testing*

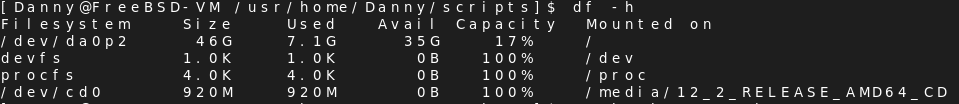
Following our test plan, we must first check whether the script produces the desired output to the user. On running the script and checking the email address supplied to the *mail* command, one can see that an email has been received that gives the user a disk space warning:



It is now necessary to check whether this information is accurate. To do this, a simple *df* command was run on the target VM:



We can see from the results that for our root drive, we have 2.8GB available which represents a total capacity in use of 93%. So far, this matches our script’s output. However, it is also important to test the script’s results when disk use is below 90%. To accomplish this the root drive on the target computer was cleared until running the *df* command produced the below result:

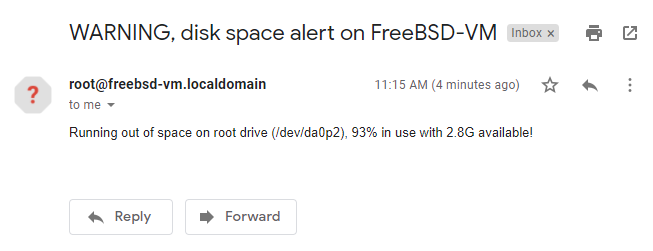


As running the script now produces no result or email, it can be concluded that shell script produces accurate results and works as intended.

Finally, we need to make sure the script is set to run on a schedule. Using the *crontab -l* we can see that the script is set to run daily at 11:15am as planned:



Furthermore, checking our inbox a day after writing the script, we can see that the script ran at the expected time and sent us an email:



# **Reference List**

Bouron, J. Chevalley, S. Lepers, B. Zwaenepoel W. Gouicem, R. Lawall, J. Muller, G and Sopena J. 2018. *The Battle of the Schedulers: FreeBSD ULE vs. Linux CFS*. 2018 USENIX Annual Technical Conference. [Online]. Available at: <https://www.usenix.org/system/files/conference/atc18/atc18-bouron.pdf> [Accessed 29th November 2020]

Canonical. *Ubuntu on public clouds.* [Online]. Available at: <https://ubuntu.com/public-cloud> [Accessed 30th November]

Dube, R. 1998. *A Comparison of the Memory Management sub-systems in FreeBSD and Linux.* [Online]. Available at: <https://drum.lib.umd.edu/bitstream/handle/1903/962/CS-TR-3929.pdf?isAllowed=y&sequence=2> [Accessed 30th November 2020]

Ferrell, J. 2020. *FreeBSD Quickstart guide for Linux Users*. Chapter 2. [Online]. Available at: [2. Default Shell (freebsd.org)](https://www.freebsd.org/doc/en_US.ISO8859-1/articles/linux-users/shells.html) [Accessed 29th November 2020]

Hickmann, B. Shook, K. 2007. *ZFS and RAID-Z: The Uber-FS?* [Online]. Available at: <http://pages.cs.wisc.edu/~remzi/Classes/736/Fall2007/Projects/BrianKynan/paper.pdf> [Accessed 30th November 2020]

Kalin, M. 2019. *CFS: Completely fair process scheduling in Linux.* [Online]. Available at: <https://opensource.com/article/19/2/fair-scheduling-linux> [Accessed 30th November 2020]

Lehey, G. 2018. *Explaining BSD.* [Online]. Available at: <https://www.freebsd.org/doc/en_US.ISO8859-1/articles/explaining-bsd/index.html> [Accessed 29th November 2020]

Looney, J. 2019. *Netflix and FreeBSD: Using Open Source to Deliver Streaming Video.* [Online]. Available at: <https://papers.freebsd.org/2019/fosdem/looney-netflix_and_freebsd/> [Accessed 30th November 2020].

Lozi, J. Lepers, B. Funston, J. Gaud, F. Quema, V. Fedorova, A. 2019. *The Linux Scheduler: A Decade of Wasted Cores*. EuroSys 2016. [Online]. Available at: <https://www.ece.ubc.ca/~sasha/papers/eurosys16-final29.pdf> [Accessed 29th November 2020]

Many contributors. 2020. *FreeBSD Handbook*. [Online]. Available at: <https://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/> [Accessed 29th November 2020]

McKusick, M., Neville-Neil, G. and Watson, R., 2015. *The Design and Implementation of the FreeBSD Operating System*. Upper Saddle River, NJ: Addison Wesley. Pp 22-23.

Roche, D. 2020. *ZFS focus on Ubuntu 20.04 LTS: what’s new?* [Online]. Available at: <https://ubuntu.com/blog/zfs-focus-on-ubuntu-20-04-lts-whats-new> [Accessed 30th November 2020]

Yang, K. and Wallace, K. 2011. *File Systems in Linux and FreeBSD: A Comparative Study.* Journal of Emerging Trends in Computing and Information Sciences. Vol. 2, No. 9. [Online]. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.458.4324&rep=rep1&type=pdf> [Accessed 30th November 2020]