Arvi Koivulehto

Installation and Testing of Google Fuchsia Operating System in Virtual Environment

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

13 June 2025

Abstract

Author:	Arvi Koivulehto
Title:	Installation and Testing of Google Fuchsia Operating System in Virtual Environment
Number of Pages:	34 pages + 3 appendices
Date:	13 June 2025
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Smart IoT Systems
Supervisors:	Keijo Länsikunnas, Senior Lecturer

The goal of the final year project was to study the installation and testing process in a virtual environment and to assess the current maturity of the Fuchsia operating system. The literature part examines first the microkernels and then specifically Fuchsia, while the practical part involves compiling, testing and measuring the Fuchsia operating system builds.

Google is developing a new modular microkernel operating system to replace their existing operating systems, such as Android, ChromeOS, WearOS and Google Cast. The open-source OS has been in development since 2017 and sees hundreds of new commits per week. It is still a work in progress, but Google has created an extensive site for developers (fuchsia.dev) which provides tutorials for various tasks within their new ecosystem.

Fuchsia is run using its own emulator, which is based on the Android emulator, which in turn is based on QEMU. Fuchsia provides an Alpine Linux container, which was used to run benchmarking tests. These results were compared to a QEMU container also running Alpine Linux. To avoid possible breaking updates during the testing period, Fuchsia's source code was downloaded only once and not updated afterwards.

Many of the tests did not work and based on those that did, it is clear that Fuchsia is not ready for non-experimental use. The pace of development is so rapid that it is hard to know where Fuchsia's capabilities currently stand. Despite the seemingly extensive documentation, many of the key aspects of the documentation are out of date or missing. Fuchsia's Linux container does not correspond to standard Linux installation and many basic programs do not work. Fuchsia is an ambitious project that still clearly requires plenty of development.

Keywords: Google, Fuchsia, Android, ChromeOS, WearOS, Cast, microkernel, operating system

Tiivistelmä

Tekijä:	Arvi Koivulehto
Otsikko:	Google Fuchsia -käyttöjärjestelmän asentaminen ja testaaminen virtuaaliympäristössä
Sivumäärä:	34 sivua + 3 liitettä
Aika:	13.6.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto–ja viestintätekniikka
Ammatillinen pääaine:	Smart IoT Systems
Ohjaajat:	Lehtori Keijo Länsikunnas

Tämä insinöörityö käsittelee Fuchsian asentamista ja testaamista virtuaaliympäristössä. Kirjallisuusosassa tarkastellaan ensin mikroytimiä ja sitten erityisesti Fuchsiaa. Kokeellisessa osassa käännetään, testataan ja mitataan Fuchsian koontiversioita.

Google on kehittämässä uutta modulaarista mikroydinpohjaista käyttöjärjestelmää, jonka tavoitteena on korvata Googlen olemassaolevat käyttöjärjestelmät Android, ChromeOS, WearOS ja Google Cast. Kehitys on aloitettu vuonna 2017, ja uusia committeja tulee satoja viikossa. Fuchsia on yhä työn alla, mutta Google on luonut sivuston (fuchsia.dev), joka tarjoaa kehittäjille paljon tietoa ja tutoriaaleja heidän uudesta ekosysteemistään.

Fuchsialla on oma emulaattorinsa, joka pohjautuu Android-emulaattoriin, joka vuorostaan pohjautuu QEMUun. Fuchsia tarjoaa Alpine Linux -kontin, jota käytettiin suorituskykytestien ajamiseen. Näiden testien tuloksia verrattiin QEMU:lla pyörivään toiseen Alpine Linux -konttiin. Mahdollisten päivityksistä johtuvien ongelmien välttämiseksi Fuchsian lähdekoodi ladattiin kerran, eikä sitä päivitetty jälkeenpäin.

Monet testit eivät toimineet. Toimivien testien tulosten perusteella Fuchsia ei ole valmis muuhun kuin testikäyttöön. Kehityksen tahti on niin nopeaa, että on vaikea arvioida, mikä Fuchsian suorituskyky nyt on. Huolimatta laajalta vaikuttavasta dokumentaatiosta monet sen keskeisistä osista ovat vanhentuneita tai puuttuvat kokonaan. Fuchsian Linux-kontti ei vastaa standardia Linux-asennusta ja monet perusohjelmista eivät toimi. Fuchsia on kunnianhimoinen projekti, joka selvästi vaatii paljon lisää kehitystä.

Avainsanat: Google, Fuchsia, Android, ChromeOS, WearOS, Cast, mikroydin, käyttöjärjestelmä

Contents

1 Introduction

Google is developing a new microkernel operating system called Fuchsia. The operating system aims to replace Google's existing operating systems ChromeOS, Android, WearOS and Google Home [1]. This effort raises several questions, mostly relating to why Google is developing a brand-new operating system from the ground up, instead of adapting one of their existing ones. This decision is surprising, especially considering that Google has a long history of adapting the Linux kernel for various purposes and that currently Android, a Google-developed Linux derivative, is the most popular operating system in the world [2].

To understand Google's reasoning, it is best to consider the differences between monolithic kernels, such as Linux, and microkernels such as Fuchsia. There is a long history of microkernel development, but so far there have been few examples of successful microkernel systems and most efforts by major companies, such as Apple and Microsoft, have ended in failure or at the very least major scaling back of their initial project goals [3].

Fuchsia was inspected based on source code downloaded on 7th February 2025. This installation has not been updated since, in order to have a coherent snapshot from a certain date. As the Fuchsia repository is updated hundreds of times per week, this snapshot is severely out of date and without insider knowledge from Google, it is hard to predict whether any given functionality has improved or declined since it was tested for this final year project. This pace of development should also answer any questions on whether Fuchsia is still under active development.

This final year project studies the installation and testing process in a virtual environment and assesses the current maturity of the Fuchsia operating system. The literature part examines first the microkernels and then specifically Fuchsia, while the practical part involves compiling, testing and measuring the Fuchsia operating system builds in a virtual environment using an old laptop computer.

2 Microkernels

Microkernel operating systems started out in the 1980s as an effort to offer more flexibility in kernel design and allow a more modular design where one could swap out different parts of the operating system, such as the filesystem without having to recompile the whole kernel. It is even possible to include multiple filesystems, depending on which one is best suited to the task. [4.]

Unfortunately, the first-generation microkernels were found to be much slower than comparable monolithic kernels and were only usable as research projects [3]. Later on in the 1990s, the second generation looked much more promising, but microkernels have still failed to gain market share [2, 4].

2.1 The status of microkernels in 2002

In 2002, Thorsten Scheuermann wrote a brief review of the state of microkernel systems. This review provides a snapshot of the state of the art in the early late 1990s and early 2000s. In his review, Scheuermann found that microkernels offer advantages in modularity and fault-tolerance over traditional monolithic kernels, and their performance limitations are a small price to pay for the gains. He, however, observed that mainstream desktops are rather conservative and correctly predicted that they would not adopt microkernels in the near future. [5.]

When conducting benchmarking, Scheuermann runs a Linux kernel on top of a Mach microkernel, which had been cited as one possible use case for microkernels, where one can swap between different kernels as needed. Unsurprisingly, he found that running Linux on top of a microkernel was much less performant than running it on its own. Naturally, it appears rather pointless to run a microkernel in order to run a monolithic kernel, when one could just run the monolithic kernel on its own. It seems that this use case has indeed rarely come up in the practical usage of microkernels and its lack of performance is perhaps unhelpful in determining the viability of microkernels as an alternative to traditional kernel architectures. [5.]

Scheuermann noted that the small footprint of microkernels would be most beneficial in embedded computing devices, and that the Linux kernel offers similar flexibility if one is willing to compile their own custom kernel. Here things developed a bit differently from his expectations as evidently the strength of the Linux ecosystem has overcome the limitations of monolithic kernels and instead Linux has taken market share from established microkernel systems, such as QNX, wherever real-time capabilities are not absolutely required. [5.]

2.2 Real-time microkernels

In his article “What is real time and why do I need it?”, Stephen Furr, a senior product manager for QNX, goes through the definition of a real-time operating system and what benefits it offers [6]. Based on the definition, Fuchsia is not a real-time operating system (RTOS), but many microkernels are, and therefore it is worthwhile to understand which capabilities RTOSs offer.

The basic definition of a real-time system is that the result of an operation is not only dependent on the computation but also on how long said computation takes. In a real-time system, there is always at least an implicit timeframe for an operation to take place and if this timeframe is breached, the operation is said to have failed. As an example of how this sort of expectation is implied even in traditional computer systems, Furr points out that no-one wants their salary to be two weeks late because the bank’s computer was slow. [6.]

Real-time systems are further divided into hard and soft real-time. In a hard real-time system a computation result that arrives late is worthless, while in a soft real-time system its value has merely diminished [6]. Usual discussions seem to imply like these systems would be mutually exclusive but there seems to be no reason why a real-time system could not handle either use case. In a modern car, if it takes a couple of seconds too long to calculate how much fuel is left, that is hardly an issue, but if the same delay happens in the braking system, the results can be fatal. Often cars in fact use multiple systems for different purposes. Especially the entertainment systems these days are more likely to be Linux-based rather than an RTOS [7].

A hard real-time operating system guarantees that operations are executed in a deterministic time frame, with the caveat that external interrupts may interfere with this execution. Furr doesn’t specify what will happen if these interrupts overwhelm

the system. Presumably there is a requirement for a failsafe mechanism to either ignore an overwhelming number of interrupts or to fail the task in a controlled manner, rather than to freeze the whole, which is the usual solution for similar issues in non-real-time systems. [6.]

Fuchsia is not a real-time operating system even though its Zircon kernel is theoretically real-time capable [8, 9]. This raises the interesting question of why Google didn't see the need for a real-time operating system as it would have been a unique selling point among the other mainstream operating systems. Moreover, this necessarily puts limits on where Fuchsia can be used. A real-time operating system has the potential to be a universal system that can be used even in life-or-death situations, such as control systems for vehicles. Without real-time capabilities, Fuchsia can only compete in sectors where Linux is already established as it is not an RTOS either. Clearly real-time operating systems are not as necessary as once thought as Linux has chased out QNX from various embedded applications that have previously been its domain, starting from iOS and Linux-based Android taking over the smartphone market from QNX's Blackberry phones. [10.]

What Google clearly has paid attention to is security [1]. They have dedicated a lot of effort to documenting their design choices and are in favour of a strong security system, at least when it comes to third party issues. Meanwhile, questions may be raised about their internal commitment to privacy, as highlighted by the Fuchsia teams reported power struggle between the business and engineering side over kernel-level telemetry shows [11]. Perhaps one of the reasons Google chose to develop a microkernel is the inherent security benefits of a microkernel system.

2.3 The security implications of kernel design

In their paper, Simon Biggs, Damon Lee, and Gernot Heiser from the University of New South Wales argue that monolithic kernel design is inherently less secure than microkernel design. Their main argument relies on the fact that as microkernels are smaller, their safety is easier to verify, and the small size also limits the number of functions that are run in privileged mode, limiting the size of the attack vector. [12.]

To support their conclusion, the writers mainly compare Linux kernel's discovered bugs and vulnerabilities to a microkernel named sel4, developed by the University of New South Wales. They claim that sel4's implementation has been formally proven correct and that its worst-case execution times have been analyzed. As evidence of the rising complexity of securing the Linux kernel, they cite how the size of Linux code base grows over time, pointing out that the number of software bugs increases as the codebase grows. The benefits of microkernels are illustrated by going through individual past Linux vulnerabilities and categorizing how sel4 or microkernels in general avoid these pitfalls. They acknowledge that Linux might not necessarily represent the kinds of vulnerabilities in monolithic kernels in general, but state that the same types of issues have been found in Windows and macOS. [12.]

Their praise of their own university's kernel raises certain doubts, as there surely has been more research interest and analysis devoted to Linux with its global userbase. The findings in the paper, however, do correspond to the theorized security gains from compartmentalization in microkernel architecture over monolithic kernels' less regimented approach [12].

2.4 Scalability in monolithic vs microkernel operating systems

The previous section described the security benefits of the intricate access control enforced by microkernels, but the research paper "Extreme high performance computing or why microkernels suck" points out that this kind of encapsulation introduces performance overhead and limits the kinds of locking mechanisms that are needed to manage the potentially hundreds of processors present in high performance computers. The researchers were surprised to discover that this kind of scaling was easier to achieve in Linux, where the lack of encapsulation might cause unforeseen issues when trying to enable the kind of scaling required for the researcher's needs. They also found that the context switching required to communicate between microkernel systems' various isolated components had a much more serious limit on scalability. [13.]

The unified address space in monolithic operating systems allows kernel components to access each other freely and therefore enables a wide variety of different ways to ensure that processor resources are always properly utilized. This freed the researchers to develop different locking methods for different

components. Microkernels limit data structures to specific subsystems while in Linux a single structure can contain data from different subsystems, and they can all be handled by a single lock. In the researchers' view, this is behind the success of Linux success which is used in a multitude of computing environments from embedded devices to supercomputers. [13]

These findings are similar to the more common observation that security and reliability measures often degrade performance in computer systems. When considering programming languages, for example, the languages that allow the lowest level actions with memory unsafe operations usually deliver the best performance. This approach, however, exposes the system to more esoteric bugs and potentially more severe issues as well as security vulnerabilities when something goes wrong. As the researchers were purely looking at scalability, these tradeoffs were not really considered in the paper.

Traditionally there has been a clear divide between applications that require special operating systems developed explicitly for embedded use and the more desktop-like environments that have been taken over by operating systems such as macOS and Windows. However, with the development of Linux and increased processing power, Linux has been moving out of its original design space towards more challenging environments and proven to be extremely adaptable.

2.5 Comparing microkernel systems to Linux in embedded environments

In 2006, Ivan Stankov and Grisha Spasov from Technical University in Sofia, Bulgaria, wrote a paper comparing a microkernel-based RTOS (QNX Neutrino) to a monolithic OS (Red Hat Embedded Linux). The paper compares both the architecture and the real-life performance of said systems. Most of the focus is on the lack of real-time capabilities in Linux and the tone implies that the writers consider it a novel idea to use a non-real-time system for embedded applications. They, however, conclude that Linux is a viable alternative to QNX when real-time is not required. They also cite the lack of licensing fees as Linux's main advantage. [14.]

The state of the embedded ecosystem has changed since Stankov and Spasov wrote their paper in 2006. It appears that back then embedded Linux was a new idea and the paper criticized Linux for poor support for embedded options and a

lack of documentation. Now, over 18 years later, embedded Linux is the default embedded solution for any system capable of running a fully featured operating system. This is the same niche that Fuchsia targets, with the added goal of planning to offer a modular operating system which can fit in spaces where even Linux is too unwieldy, while still not offering real-time capabilities. Google seems to calculate that real-time is not needed as often as Stankov and Spasov assumed.

While Linux has gained popularity, QNX has been chased out of the mobile market and even the embedded market by Linux. Other RTOSs, such FreeRTOS, are still popular but only in environments where Linux cannot be run. As the name indicates, FreeRTOS shares Linux's advantage of zero licensing fees.

While QNX was a notably successful microkernel in the past, multiple companies have tried to develop their own microkernels. Both Apple and Microsoft did this in the 1990s, when they were planning on what their operating systems would look like in the new millennium. Both also ended up compromising the microkernel vision, eventually ending up with so called hybrid kernels, combining aspects of both micro- and monolithic kernels. The most similar project to Fuchsia and its aims was, however, IBM's ill-fated Workplace OS.

2.6 IBM's failed microkernel operating system

In 1991, IBM decided to develop a new microkernel operating system, Workplace OS, with the goal of using a single operating system for their various computers, ranging from PDAs to mainframes. This was supposed to simplify development and software maintenance. IBM spent 2 billion dollars, or 0.6% of their five-year revenue, on this cost-saving endeavour that would set a new direction for the company. The project was cancelled in October 1995. [3.]

The Workplace OS ran into both cost and schedule overruns. It experienced feature creep and tried to be everything for everyone. The different use cases for the operating system were planned to be implemented using "personalities" which were supposed to correspond to pre-existing operating systems such as OS/2, DOS and UNIX. The goal for this was to reduce the learning curve for the new operating system and encourage adoption by providing a familiar environment for the users. This personality system was one of the more significant pitfalls of the project as the various personalities proved difficult to implement and coordinate between the development teams. The feature was also considered too late in the

development cycle and would struggle with the limitations of the underlying microkernel design. [3.]

This IBM project from the early 1990s has obvious parallels to Google's Fuchsia project. The underlying logic is similar, and the goals are equally ambitious. Google has, however, avoided tying equally many resources to the project and has a considerably narrower product line to consider. The company is also not trying to reimplement existing operating systems although it has made similar efforts to encourage adoption by allowing Android and Flutter applications to be run in Fuchsia. This raises questions about how well developers at Google are aware of this previous failed effort by another large technology company to transition to a microkernel system.

3 Fuchsia

Fuchsia is such a new system that so far there are few academic texts written about it. Apart from the few that do exist, one must rely on articles published on technology-focused websites and in magazines. Fuchsia is not built completely from scratch but is instead built on the foundations of an existing kernel and greatly expanded upon, which provides a natural starting point for examining Fuchsia.

3.1 Zircon microkernel

Zircon microkernel is the core around which Fuchsia is built. It is based on littlekernel (LK), a real-time operating system used in embedded devices, which is also used in Android's bootloader [15]. As Zircon is made to serve a more complex operating system with more available system resources, it may have different operating requirements from littlekernel [15]. Perhaps the most notable example is that Fuchsia is not a real-time operating system so Zircon will likely not support real-time capabilities either.

LK treats all code as trusted and everything is run in privileged mode, but Zircon has memory protection features for running untrusted user applications. Zircon implements memory protection with capability-based security where users hold capabilities to kernel objects, and the capabilities determine the user's access rights. This allows for more precise security measures compared to a traditional access control list model, where every different level of access requires its own group and the categories of access can be rather limited. For example, in Linux systems, access is only defined as read, write or execute privileges which apply to every file in the system, no matter the actual purpose of the file. If a person has write access, they can edit and even completely remove a file. [15.]

3.2 Fuchsia on Google smart home devices

Google is already using Fuchsia on its Nest smart home devices but seemingly the goal has just been to reimplement past functionality and actually downplay the fundamental changes happening on the backend. From the user's point of view, the difference is practically indistinguishable and the only way to see a change is

by checking the “About device” page in the settings, the Google Home app or using an external website to see the name of the user agent. Despite this, the settings and Google Home only reveal an updated version number where Fuchsia is not mentioned, and it is only in the user agent where one can see Fuchsia written in plain text. [16.]

It seems Google is very hesitant to publicize Fuchsia at this stage, as can be seen from the fact that they have hidden information about it multiple times even when it had already been accidentally published [17] and when media outlets like Ars Technica published previews of Fuchsia’s GUI [18]. Subsequently the code necessary for running a graphical user interface was pulled from the public repositories [19].

4 Testing

Chapter 4 contains the practical part of the final year project. All tests and failed testing efforts described here were done during the final year project.

To test Fuchsia's current capabilities, Fuchsia's source code was downloaded from fuchsia.dev and built. Fuchsia could be installed on actual hardware, but the official guides [20] instructed to emulate Fuchsia inside a Linux installation and said method was chosen for this project.

Downloading the source code took almost three hours. This was mainly due to the old ThinkPad 13 Gen 2 laptop with which this final year project work was started. Using Fuchsia required getting acquainted with Google's in-house development tools and the documentation, which is not sufficiently detailed, at least outside of Google's intranet.

For the terminal shell, fish was used. It had been chosen before the project for its ease of use, but it seemed like a switch might be necessary because nothing in the documentation indicated that Fuchsia's helper scripts would work with fish. However, on further inspection, it turned out that the helper script files also exist for fish.

The software version that was used is 2025-02-07T07:02:33+00:00. As the update process is generally fragile, it was decided that updates would be avoided while testing out the software.

4.1 Fuchsia's tooling

Fuchsia uses many specialized tools developed by Google. The source code consists of multiple different git repositories which are managed using `jiri`. The fact that `jiri` manages multiple repositories creates complexities to the update process and updating the repository should be done with care as it is possible to arrive at a state where some parts of the source code have been updated but others not. The process also creates ambiguous printouts which leave one unsure whether the process has been successful or something has gone wrong.

Fuchsia was built using the `fx` tool which seemed to be an adapted version of QEMU with some extra functionality added. `fx` is the main way of engaging with the emulator and was used to both build and run the emulator as well as alter the emulator parameters.

Initially, the Fuchsia source code was downloaded by simply using curl and once that had been done, one was required to run commands to add the text `.jiri_root/bin` to PATH and `scripts/fx-env.sh` to source [21].

4.2 Building Fuchsia

Fuchsia has different build targets available, ranging from a minimal build to a ”kitchen_sink” which includes various tools and tests. There is a warning about size requirements for this target:

> //bundles/kitchen_sink is a target that causes all other build targets to be included. It is useful when testing the impact of core changes, or when making large scale changes in the code base. It also may be a fun configuration for enthusiasts to play with, as it includes all software available in the source tree. Note that kitchen sink will produce more than 20GB of build artifacts and requires at least 2GB of storage on the target device (size estimates from Q1/2019). [22.]

In reality, even the minimal build required over 50 times more disk space. For the first attempt, a 128GB hard drive was not enough. An additional difficulty was created by the fact that changing the build directory did not seem to work, despite following instructions to the letter, so both the source code and the build directory had to reside on the same hard drive.

Finishing the build required a larger hard drive (a 1 TB drive was used) even though after the build the disk usage was only 109 GB. Previously, when the build reached this point, it froze and would not progress further. Presumably some temporary files required additional disk space.

According to the terminal printout, the build process took 9 hours, 18 minutes. Fuchsia was built with a dual-core Lenovo 13 Gen 2 laptop from 2017 using an external SSD.

4.3 Testing the emulator

Starting the emulator was not quite as straightforward as might be expected. The instructions for starting the emulator seemed to assume that Fuchsia is installed natively on a separate device and not emulated even though the installation instructions are specifically for the emulator [23]. Part of the issue might have been that the fx-env.fish script looks to actually be a subset of the fx-env.sh script, instead of a full replacement. The implementation is likely WIP and perhaps that is why the option of using the fish shell was not documented.

4.4 Starting the emulator

The emulator was started using the command $\mathrm{ffx\ emu\ start}$. The Fuchsia tutorial instructed to start $\mathrm{fx\ serve}$ for ”serving Fuchsia packages” before the emulator but this produced no results as the process would error out because it could not find the emulator before it had been started. The emulator command itself took several minutes to start the first time it was run and produced no output in the meantime, leading an impatient user to assume that this command had encountered an error too. Subsequently, the emulator took approximately half a minute to start. The emulator was clearly based on the Android emulator both in terms of looks and the fact that it still contained sample text referencing Android.

There was an issue where fx serve errored out due to “expired metadata”. It was unclear to which data this error message refers to and how to update said data. Updating the project through jiri update seemingly did nothing and jiri provided no output except a notice about starting the update process and a familiar warning about not being logged into Google’s intranet.

Listing 1 is an example of which kind of warnings, bug and error messages and hints resulted when the actual failure reason was the expired metadata. The correct remedy for the issue was to refresh the source code.

```
/r/m/k/f/fuchsia[1] ~> ffx target list
NAME                SERIAL      TYPE                  STATE      ADDRS/IP             RCS
fuchsia-emulator    <unknown>   workbench_eng.x64     Product    [127.0.0.1:44843]    Y
/r/m/k/f/fuchsia ~> fx serve
WARNING: Please opt in or out of fx metrics collection.
You will receive this warning until an option is selected.
To check what data we collect, run `fx metrics`
To opt in or out, run `fx metrics <enable|disable>

2025-02-04 20:36:40 [serve] Discovery...
WARNING:
WARNING: No default device set. If 'fx serve' gets stuck on
WARNING: "Discovery..." you can set a default device by running
WARNING: 'ffx target list' to get a list of devices, and then running
WARNING: 'ffx target default set'.
WARNING:

2025-02-04 20:36:45 [serve] Device up
2025-02-04 20:36:45 [serve] Starting repository server
Switching log file to "/home/kujen/.local/share/Fuchsia/ffx/cache/logs/
    repo_default.log"
BUG: An internal command error occurred.
Error: updating the repository metadata
    1. metadata timestamp expired at 2024-12-07 02:20:34 UTC, it is now
        2025-02-04 18:36:46.321146178 UTC
More information may be available in ffx host logs in directory:
    /home/kujen/.local/share/Fuchsia/ffx/cache/logs
/r/m/k/f/fuchsia ~>

  /r/m/k/f/fuchsia ~> jiri update
  [09:34:56.997] Updating all projects
  [09:40:10.632] WARN: Some packages are skipped by cipd due to lack of access, you
      might want to run "/run/media/kujen/fuchsia_drive/fuchsia/.jiri_root/ bin/cipd
      auth-login" and try again
  /r/m/k/f/fuchsia ~>
```

Listing 1. Example of Fuchsia messages due to expired metadata.

4.5 State of the emulator

The biggest obstacle to running the Fuchsia emulator was the lack of documentation. fuchsia.dev did contain plenty of documents but it did not have a complete tutorial on starting to use Fuchsia. Instructions for the very basics existed but even these were terse and seemingly outdated. A suitable illustration was the fact that an explanation of how to run the hello world example had been provided, but there was no explanation for how the command functions, as it was not as simple as running the command `python hello_world.py`. Instead, it was necessary to write the lines of Listing 2.

```
ffx component run
core/ffx-laboratory:hello-world-cpp
fuchsia-pkg://fuchsia.com/hello-world-cpp#meta/hello-world-cpp.cm
```

Listing 2. Running the hello world in Fuchsia.

It was possible to find documentation for the structure of Fuchsia components. The documentation explained some of the syntax but the provided examples did not seem to follow the same pattern and, in fact, some of the examples seemed to be unfinished with the build file existing but the source files missing. There were instructions for writing new components for various programming languages, such as C++ and Rust to run them on Fuchsia. Before writing such a component, it was unclear whether the instructions were in fact complete, or if documentation was lacking for a crucial step in the process.

fuchsia.dev instructs that the `ffx repository list` command prints the available software packages in an ASCII table as seen in Listing 3.

```
+-------------------------+------+       +

| NAME                    | TYPE | EXTRA                 |
+=========================+======+=======================+
| minimal-packages  | pm     |  /home/alice/.local/share/Fuchsia/.../packages
|
+-------------------------+------+ ----------------------+
```

Listing 3. An ASCII table according to documentation.

However, in practice, when this command was run, it resulted in Listing 4 with no packages being shown, even though fx serve was on and able to serve packages.

```
johnson@suze /r/m/j/f/fuchsia ((99bbbe95)) [1]> ffx repository list

johnson@suze /r/m/j/f/fuchsia ((99bbbe95))>
```

Listing 4. No packages shown.

This kind of anomaly was typical when following Fuchsia's documentation during the testing phase of this final year project.

4.6 Fuchsia shell

The Fuchsia emulator offers a Dash shell environment, with POSIX commands such as ls, pwd and less. This environment can be accessed with the fx shell command, and it allowed one to examine the internal organization of Fuchsia directories.

The Fuchsia subdirectories of the / directory are shown in Listing 5.

```
$ ls -lah
dr-x------ 1 0      0          0 Jan 01 1970 .
dr-x------ 0 0      0          0 Jan 01 1970 bin
drwx------ 2 0      0          0 Feb 14 03:01 blob
dr-x------ 0 0      0          0 Jan 01 1970 boot
dr-x------ 0 0      0          0 Jan 01 1970 boot-bin
dr-x------ 1 0      0          0 Jan 01 1970 config
drwx------ 5 0      0          0 Mar 05 07:05 data
drwx------ 0 0      0          0 Jan 01 1970 dev
dr-x------ 0 0      0          0 Jan 01 1970 mnt
dr-x------ 0 0      0          0 Jan 01 1970 pkg
dr-x------ 0 0      0          0 Jan 01 1970 pkgfs
dr-x------ 0 0      0          0 Jan 01 1970 svc
dr-x------ 0 0      0          0 Jan 01 1970 system
drwx------ 4 0      0          0 Mar 05 07:05 tmp
```

Listing 5. Internal organization of Fuchsia directories.

4.7 Difficulties in measuring Fuchsia

The Fuchsia shell offers a POSIX-like environment and several POSIX commands, but not all commands that exist. For example, time commands, used to measure execution time, are missing, making it harder to evaluate the performance of the emulator.

The command in Listing 6 would measure how long it takes to run a command to read the files in the directory test_files, looking for the string “non-existent”.

```
\$ time grep -r "non-existent" test_files/

/boot/bin/sh: 2: Cannot create child process: -1 (ZX_ERR_INTERNAL): failed to
    resolve fuchsia-pkg://fuchsia.com/time\#bin/time: -1 (ZX_ERR_INTERNAL)
```

Listing 6. Time command is missing.

As GNU Compiler Collection has not been ported to Fuchsia, compiling existing Linux benchmarking software requires manual intervention and further understanding of the Fuchsia packaging process. Fuchsia offers a Linux compatibility system called Starnix, which is supposed to allow running unaltered Linux programs, but unfortunately the documentation is presented in a confusing manner. There are multiple different locations for it with similar names and with completely different contents. main:src/starnix/README.md is practically empty, while main:src/starnix/kernel/README.md has what one would expect from a README. On one of Starnix's pages on fuchsia.dev, concepts/starnix, the link to Starnix's readme pointed to a non-existent location when it was visited the first time. Later it pointed to the latter readme file even though the history section implied that no changes have been made to the files themselves. It was unclear whether the initial omission was a bug or if the file had suddenly been made publicly available.

Assuming that Starnix can be used for benchmarking software, something like the Flexible I/O tester fio, originally written for Linux kernel testing, would be ideal to test the emulator.

4.8 Setting up QEMU and Starnix

Fuchsia's Starnix system looks to be a rather straightforward Linux container, as it even contains a Dockerfile. By default, both the containers for Alpine and Debian are included. The Alpine container was chosen for this test because it was lighter but considering Alpine's quirks, the Debian container might be preferable. Starnix does not exist persistently and will be created from scratch each time the component is started as shown in Listing 7.

```
/r/m/j/f/ fuchsiaffx emu start ((99bbbe95)|?) 15:36
Auto resolving networking to user-mode. For more information see https://fuchsia.
    dev/fuchsia-src/development/build/emulator\#networking
Logging to "/home/johnson/.local/share/Fuchsia/ffx/emu/instances/fuchsia-emulator
    /emulator.log"
Waiting for Fuchsia to start (up to 60 seconds)..
Emulator is ready.
/r/m/j/f/fuchsia [1]ffx component run /core/starnix_runner/playground:alpine
    fuchsia-pkg://fuchsia.com/alpine\#meta/alpine_container.cm
URL: fuchsia-pkg://fuchsia.com/alpine\#meta/alpine_container.cm
Moniker: core/starnix_runner/playground:alpine
Creating component instance...
Resolving component instance...
```

```
Starting component instance...
Component instance is running!
/r/m/j/f/ fuchsiaffx starnix console -m /core/starnix_runner/playground:alpine /
    bin/sh 2.457s ((99bbbe95)|?) 15:38
/ \#
```

Listing 7. Starting the Starnix emulator.

Setting up the QEMU virtual machine proved to be a Sisyphean task, thanks to the very inadequate tooling and Alpine's bizarre decision to exclude the uinput module from their image that is supposedly purpose-built for virtual machines. Because this module was excluded, one could not follow the normal process to set up clipboard sharing between the QEMU host and guest machines. Instead of this blunder being documented at the start of Alpine's instructions for Spice Agent (which allows clipboard sharing), it was documented at the very end of Alpine's wiki page https://wiki.alpinelinux.org/wiki/How_to_run_Spice_Agent. The provided command did not even work. The process was especially rueful when trying to follow general instructions for Spice Agent, instead of following Alpine's own instructions, as the normal starting point was that the software would work similarly no matter the distribution and that following the manual is preferable to following a subpar wiki page.

It has been difficult to find ready-made benchmarking scripts instead of simply benchmarking tools where custom scripts can be executed. Custom test scripts were written but it turned out grep works differently in Alpine's sh rather than on sh on different Linux distributions. On other distributions, `grep -r "test"` recursively went through all the files in a directory and searched for the word "test". On Alpine this was not the case. Therefore, further investigation is needed to fix the custom read script.

The custom write script of Listing 8 written for this final year project works and allows to compare the execution speed of both containers. The outcome was that it takes considerably longer to run in the Starnix container compared to the Alpine container.

```
#!/bin/sh

# Define the directory name
DIR="test_files"

# Check if the directory
exists if [ ! -d "$DIR" ];
then
# Create the directory if it does not exist
mkdir "$DIR"
    echo "Directory '$DIR'
created." else
echo "Directory '$DIR' already exists."
fi

# Loop to create 10,000 files
NUMBER=10000
CONTENT=test
for i in $(seq $NUMBER); do
    echo "$CONTENT" > "$DIR/file_$i.txt"
done

echo "Created $NUMBER files with the content '$CONTENT' in the '$DIR' directory."
```

Listing 8. A custom write script to create 10,000 files.

Using the time command, if the directory did not exist, this took about 1.65 seconds on Starnix, and if the directory did exist, then the time was reduced to about 1 second.

On the QEMU Alpine container, there was a peculiar discrepancy where, if the directory did not exist, the script took approximately 0.17 seconds but 0.28 if the directory was already there. This was the opposite result from the Starnix container. The difference might have been due to slight differences in the script file because it had proven difficult to transfer files to the QEMU container.

Later, a test suite by a Linux website called Phoronix was discovered and used for testing.

4.9 Phoronix tests

Phoronix tests worked on QEMU Alpine after significant difficulties and the results were retrieved using SSH as shown in Listing 9.

```
alpine8-test-1:

    Processor: 12 x AMD Ryzen 5 7600X 6-Core (12 Cores), Motherboard: QEMU
        Standard PC (Q35 + ICH9 2009) (rel-1.16.3-2-gc13ff2cd-prebuilt.qemu.
        org BIOS), Memory: 30GB, Disk: 15GB, Graphics: bochs-drmdrmfb, Monitor
        : QEMU Monitor

    OS: Alpine Linux v3.21 3.21.3, Kernel: 6.12.13-0-lts (x86_64), Compiler:
        GCC 14.2.0, File-System: tmpfs, Screen Resolution: 1280x800


  Flexible IO Tester 3.36
  Type: Random Read - Engine: IO_uring - Direct: No - Block Size: 4KB - Job
      Count: 1 - Disk Target: Default Test Directory
  MB/s > Higher Is Better
  alpine8-test-1 . 2611 |=====================================|


  Flexible IO Tester 3.36
  Type: Random Read - Engine: IO_uring - Direct: No - Block Size: 4KB - Job
      Count: 1 - Disk Target: Default Test Directory
  IOPS > Higher Is Better
  alpine8-test-1 . 668333 |=====================================|


  Flexible IO Tester 3.36


  Type: Random Write - Engine: IO_uring - Direct: No - Block Size: 4KB - Job
      Count: 1 - Disk Target: Default Test Directory
  MB/s > Higher Is Better
  alpine8-test-1 . 2485 |=====================================|


  Flexible IO Tester 3.36
  Type: Random Write - Engine: IO_uring - Direct: No - Block Size: 4KB - Job
      Count: 1 - Disk Target: Default Test Directory
  IOPS > Higher Is Better
  alpine8-test-1 . 636200 |=====================================|
```

Listing 9. Results of Phoronix test suite.

The binary also worked in Starnix containers but required additional dependencies. Copying the necessary files was time-consuming and involved some iteration. Initial results requesting additional software components are shown in Listing 10.

```
/ # ./tmp/phoronix-test-suite
./tmp/phoronix-test-suite: cd: line 54: can't cd to /usr/share/phoronix-test-
    suite: No such file or directory

PHP must be installed for the Phoronix Test Suite
The PHP command-line package is commonly called php-cli, php7-cli, php8-cli, or
    php.
For more information, see the included documentation or online via:
        https://github.com/phoronix-test-suite/phoronix-test-suite/tree/master/
            documentation

The command to likely run for your operating system is:
# apk add php7 php7-dom php7-zip php7-json php7-simplexml
```

Listing 10. Initial results of the Phoronix test suite in a Starnix container.

Listing 11 looks like the wildcard operator would have worked but in fact it did not provide the intended results.

```
johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> source scripts/fx-env.fish
johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> ffx component copy /home/johnson/
    git/bachelors-thesis/test/write.sh core/starnix_runner/playground:alpine::
    out::fs_root/tmp
johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> ffx component copy /home/johnson/
    git/bachelors-thesis/test/phoronix/phoronix-test-suite core/starnix_runner/
    playground:alpine::out::fs_root/tmp
johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> ffx component copy /home/johnson/
    git/bachelors-thesis/test/phoronix/* core/starnix_runner/playground:alpine
    ::out::fs_root/tmp


johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> ffx component copy /home/johnson/
    git/bachelors-thesis/test/phoronix/* core/starnix_runner/playground:alpine
    ::out::fs_root/tmp
johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> bash
johnson@suze:/run/media/johnson/fuchsia_drive/fuchsia> ffx component copy /home
    /johnson/git/bachelors-thesis/test/phoronix/* core/starnix_runner/
    playground:alpine::out::fs_root/tmp
johnson@suze:/run/media/johnson/fuchsia_drive/fuchsia>
```

Listing 11. An attempt to copy the Phoronix test suite into a Starnix container.

This issue is compounded by the Starnix Alpine container not recognizing certain standard Alpine commands such as openrc for setting up sshd or setup-alpine which is the standard setup script for Alpine.

4.10 Comparing QEMU and Starnix performance

There were initial issues running benchmarks in Starnix's Alpine container, as the normal method of installing software for Alpine wasn't functional and Fuchsia's method for transferring files recursively between the host machine and the emulator hadn't yet been implemented.

Starnix already offered two different variants of the Linux container, one for Alpine and one for Debian. The Debian container was non-functional in the tested software version. Listings 12 shows the setup of a Debian container, Listing 13 shows how it was seemingly running OK and Listing 14 shows how it actually did not work at all.

```
johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> source scripts/fx-env.fish
johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> ffx component run /core/
    starnix_runner/playground:debian fuchsia-pkg://fuchsia.com/debian#meta/
    debian_container.cm
^ C

johnson@suze /r/m/j/f/fuchsia ((99bbbe95)) [SIGINT]> ffx emu start
Auto resolving networking to user-mode. For more information see https://fuchsia.
    dev/fuchsia-src/development/build/emulator#networking
Logging to "/home/johnson/.local/share/Fuchsia/ffx/emu/instances/fuchsia-emulator
    /emulator.log"
Waiting for Fuchsia to start (up to 60 seconds)..
Emulator is ready.
```

Listing 12. Setting up a Debian container.

```
johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> fx serve &
johnson@suze /r/m/j/f/fuchsia ((99bbbe95))> WARNING: Please opt in or out of fx
    metrics collection.


You will receive this warning until an option is selected.
To check what data we collect, run `fx metrics`
To opt in or out, run `fx metrics <enable|disable>

2025-04-03 21:44:25 [serve] Discovery...
WARNING:
WARNING: No default device set. If 'fx serve' gets stuck on
WARNING: "Discovery..." you can set a default device by running
WARNING: 'ffx target list' to get a list of devices, and then running
```

```
WARNING: 'ffx target default set'.
WARNING:

2025-04-03 21:44:28 [serve] Device up
2025-04-03 21:44:28 [serve] Starting repository server
Switching log file to "/home/johnson/.local/share/Fuchsia/ffx/cache/logs/
    repo_workbench-eng.log"
Serving repository '/run/media/johnson/fuchsia_drive/fuchsia/out/workbench_eng.
    x64/amber-files' over address '[::]:8083'.
fx serve &                              ffx component run /core/starnix_runner/
    playground:debian  fuchsia-pkg://fuchsia.com/debian#meta/debian_container.cm
URL: fuchsia-pkg://fuchsia.com/debian#meta/debian_container.cm
Moniker: core/starnix_runner/playground:debian
Creating component instance...
Resolving component instance...
Starting component instance...
Component instance is running!
```

Listing 13. A Debian container looked like it was starting as intended.

```
WARNING: your component may not run correctly due to some required capabilities
    not being available:

       Used Capability                          Result
[] fuchsia.element.GraphicalPresenter (Optional) `fuchsia.element.
    GraphicalPresenter` was not offered to
                                                `core/starnix_runner/playground:
                                                    debian` by parent
[]  fuchsia.logger.LogSink                      Success
[] data                                         `data` was not offered to `core/
    starnix_runner/playground:debian` by parent
[]  starnix                                     Success

       Exposed Capability                 Result
[] fs_root                                Success
[] fuchsia.component.runner.ComponentRunner Success
[] fuchsia.memory.attribution.Provider    Success
[] fuchsia.starnix.container.Controller   Success
[] starnix_container
                                          Success
For further diagnosis, try:

 $ ffx component route core/starnix_runner/playground:debian fuchsia.element.
    GraphicalPresenter
 $ ffx component route core/starnix_runner/playground:debian data
```

Listing 14. A Debian container failed to run.

Despite Starnix offering an Alpine container, which even came with a Dockerfile, it was not a standard Alpine image. It had its own version of the Linux kernel and many Linux programs ran into persistent errors, including certain rather essential ones, such as git as shown in Listing 15. In order to install new software, one had to add a nameserver to /etc/resolv.conf. Simply, for example, adding the line "nameserver 8.8.8.8" was sufficient to enable the use of apk.

```
/tmp # git clone https://github.com/axboe/fio
Cloning into 'fio'...
error: cannot create async thread: Resource temporarily unavailable
fatal: fetch-pack: unable to fork off sideband demultiplexer
```

Listing 15. Git is permanently unusable in the Starnix Alpine container despite the claim of temporary unavailability.

Listings 16 and 17 show the abridged system information for QEMU and Starnix Alpine containers. Appendix 1 shows the full system information.

```
test@alpine9 ~/shellbench (master)> inxi -Fxz System:
  Kernel: 6.6.14-0-lts arch: x86_64 bits: 64 compiler: gcc v: 13.2.1
    Console: pty pts/0 Distro: Alpine Linux v3.19
CPU:
  Info: 4x 1-core model: AMD Ryzen 5 7600X bits: 64 type: SMP arch: Zen 4
    rev: 2 cache: L1: 4x 128 KiB (512 KiB) L2: 4x 512 KiB (2 MiB)
    L3: 4x 16 MiB (64 MiB)
  Speed (MHz): avg: 4691 min/max: N/A cores: 1: 4691 2: 4691 3: 4691 4: 4691
    bogomips: 37545
  Flags: avx avx2 lm nx pae sse sse2 sse3 sse4_1 sse4_2 sse4a ssse3 svm
Info:
  Processes: 162 Uptime: 4m Memory: total: 8 GiB available: 7.76 GiB
  used: 464.4 MiB (5.8%) Init: SysVinit rc: OpenRC v: 0.52.1 runlevel: default
  Compilers: N/A Packages: 62 Shell: ash (busybox) inxi: 3.3.31
```

Listing 16. QEMU Alpine container's abridged system information.

```
/tmp/shellbench-master # inxi -Fxz System:
  Kernel: 5.10.199-starnix arch: x86_64 bits: 64 compiler: clang v: HEAD
    Console: pty pts/0 Distro: Alpine Linux v3.19
CPU:
  Info: quad core model: QEMU Virtual CPU bits: 64 type: MCP arch: N/A
    cache: N/A
  Speed: N/A min/max: N/A cores: No per core speed data found. bogomips: N/A
  Flags: N/A
Info:
  Processes: 6 Uptime: 5d 4h 44m Memory: total: N/A available: 8 GiB
  used: 3.83 GiB (47.9%) Init: SysVinit Compilers: gcc: 13.2.1 Packages: 115
  Shell: ash (busybox) inxi: 3.3.31
```

Listing 17: Starnix Alpine container's abridged system information.

Various benchmarking software were tried, such as Phoronix, fio, sysbench. Even regular programs with their own benchmarking utilities such as 7z and cryptsetup were tried unsuccessfully in Starnix as seen in Listings 18 and 19. All these tests passed successfully in the QEMU Alpine container. Arch Linux's wiki was utilized to get a quick overview of possible benchmarks. With programs such as Phoronix, even the download and installation process took a considerable amount of time in Fuchsia, sometimes half an hour or more [24]. Some of the failed test results are listed in Appendix 2. Finally, the program that did work was shellbench. It is intended for comparing different POSIX shells and part of a larger suite of tools developed for testing shell scripts [24, 25].

```
/ # 7z b

7-Zip (z) 23.01 (x64) : Copyright (c) 1999-2023 Igor Pavlov : 2023-06-
20 64-bit locale=C.UTF-8 Threads:4 OPEN_MAX:1024

Compiler: 13.2.1 20231014 GCC 13.2.1: SSE2
Linux : 5.10.199-starnix : 2025-02-07T07:02:33+00:00 : x86_64 : KVMKVMKVM : [FB]~
: 0.0.0.0.0.0
PageSize:4KB hwcap:0
AMD Ryzen 5 7600X 6-Core Processor (A60F12)

1T CPU Freq (MHz): 4653 4791 4773 4781 4774 5105 5373
2T CPU Freq (MHz):

System ERROR:
errno=11 : Resource temporarily unavailable
```

Listing 18. The unknown unavailable resource never became available.

```
/tmp # cryptsetup benchmark
# Tests are approximate using memory only (no storage IO).
PBKDF2-sha1 2191381 iterations per second for 256-bit key
PBKDF2-sha256 3584875 iterations per second for 256-bit key
PBKDF2-sha512 1762312 iterations per second for 256-bit key
PBKDF2-ripemd160 1073261 iterations per second for 256-bit key
PBKDF2-whirlpool 821768 iterations per second for 256-bit key
argon2i N/A
argon2id N/A
Required kernel crypto interface not available.
Ensure you have algif_skcipher kernel module
loaded.
```

Listing 19. Cryptsetup is missing both argon2i and argon2id and kernel features.

Trying to run fio errored out on shm segment setup.

Sysbench did not work either as shown in Listing 20.

```
FATAL: sb_thread_create() for thread #0 failed. errno = 0 (No error information)
```

Listing 20. Sysbench seemingly failed to create a thread.

Thankfully shellbench worked. It has 12 different sample scripts that are used to test various features in different shells. Not all these features are present in all shells or their implementations and are shown as errors in the test log. The names of the scripts and their subsections describe the functionality being tested. For example, assign.sh tests various ways of variable assignments and stringops.sh different features for string manipulation, such as splitting strings into substrings or sorting strings according to their length.

Direct assignment just tests out assigning a value to a variable and unsurprisingly showed no differences between containers. The eval assignment uses the built-in eval function to assign a variable and here the sh implementations were practically identical but there was a notable difference between them in bash performance. The largest difference occured when trying to assign a variable using command substitution. Both shells showed a major slowdown on Starnix.

As can be seen in Table 1, there was a considerable performance difference between the different containers depending on the operation, despite both having the same number of cores and RAM. The tests for the null script, in fact, occasionally ran marginally faster on Fuchsia, but while the substring operations were measured in thousands of executions per second on QEMU, they reached mere hundreds on Fuchsia.

Table 1: QEMU Alpine and Fuchsia Starnix Alpine container shellbench selected results (executions per second)

name	QEMU sh	Starnix sh	QEMU sh / Starnix sh	QEMU bash	Starnix bash	QEMU bash / Starnix bash
eval.sh: direct assign	1,142,275	1,144,476	99.8%	278,549	249,909	111.5%
eval.sh: eval assign	666,838	132,362	96.1%	693,713	88,353	149.8%
eval.sh: com-mand subs	8,537	4,273	879.2%	971	732	583.7%

5 Conclusions

The goal of the final year project was to study the installation and testing process in a virtual environment and to assess the current maturity of the Fuchsia operating system.

Google's Fuchsia is an ambitious endeavour to try and develop a single operating system that would fulfil all of Google's needs. Because the intention is to create a modular system that replaces Google's existing solutions, the system has been developed without real-time capabilities as Google is not currently targeting that market. If the modularity is implemented suitably well, RTOS capabilities could in theory be added later, but such changes at the core of the kernel would likely require changes in other components that have been built without accounting for RTOS constraints.

Fuchsia is still clearly a work in progress and measuring its current capabilities has proved difficult. Despite numerous documents and a whole site dedicated to its development, a significant amount of reverse engineering is still needed to understand and analyze Fuchsia thoroughly. The documentation looks comprehensive at first glance, but the project does not fully match its documentation and key parts of required instructions are missing. It looks like the documentation serves partly as notes for Google's developers who have inside knowledge of the project.

There have been numerous attempts by major corporations to develop a mainstream microkernel operating system, targeting general computing and not just embedded applications. There is a surprising lack of research into what has resulted in the continuing dominance of traditional monolithic kernels over microkernels and the lack of success despite the ambitious predictions. One can find papers on the individual attempts, but a more comprehensive history of microkernels would be beneficial and illustrative of the larger trends in this field.

Similarly, the transition away from real-time operating systems towards more relaxed processor time management schemes has been perhaps surprising, considering how in the past the capabilities were viewed as much more essential. This can in large part be explained by the increased processing power making non-real-time systems work in a timely manner even in embedded applications.

However, this development reinforces the need to maintain multiple different operating systems working in conjunction when it comes to, for example, computing in vehicles. RTOSs are absolutely required for controlling the vehicle movement while the more robust ecosystems developed for Android and iOS make these mobile operating systems more appealing for entertainment and navigational tasks.

The potential of microkernel and real-time systems remains elusive, as does the ambition of having a unified operating system covering all computing applications. The current software infrastructure is built on foundations laid decades ago, when the idea of ubiquitous computing was in the realm of science fiction instead of an everyday reality. Modern concerns for cybersecurity and privacy were not considered thoroughly back then which leaves certain vulnerabilities but also means that certain anti-features such as user surveillance are absent on the kernel level. With the development of new operating systems, remote monitoring can more easily find its way into the very core of future OSes.

References

1 Pagano, Francesco; Verderame, Luca & Merlo, Alessio. 2021. “Understanding Fuchsia Security”. In: arXiv preprint arXiv:2108.04183. <https://arxiv.org/abs/2108.04183>. Visited 5 June 2024.

2 Statcounter.com. May 2024. Operating System Market Share Worldwide. Online. <https://gs.statcounter.com/os-market-share>. Visited 5 June 2024.

3 Fleisch, Brett D & Co, Mark Allan A. 1998. “Workplace Microkernel and OS: a Case Study”. In: Software: Practice and Experience 28.6, pp. 569–591. Visited 5 June 2024.

4 Liedtke, Jochen. September 1996. “Toward Real Microkernels”. In: Commun. ACM 39.9, pp. 70–77. <https://doi.org/10.1145/234215.234473>. Visited 5 June 2024.

5 Scheuermann, Thorsten. 2002. “Evolution in Microkernel Design”. In: Computer Science Department, University of North Carolina, Chapel Hill, NC. <https://u.cs.biu.ac.il/~wisemay/2os/microkernels/scheuermann.pdf>. Visited 5 June 2024.

6 Furr, Steve. 2002. “What Is Real Time and Why Do I Need It?” In: QNX Software Systems Ltd, pp. 12–5. <https://www.researchgate.net/publication/265492079_What_is_real_time_and_why_do_I_need_it>. Visited 30 April 2025.

7 Brown, Eric. July 2013. “The Rise of Linux in In-vehicle Infotainment (IVI)”. <https://linuxgizmos.com/linux-based-in-vehicle-infotainment-on-the-rise/>. Visited 30 April 2025.

8 Little Kernel. April 2025. <https://github.com/littlekernel/lk>. Visited 30 April 2025.

9 Fuchsia.dev Glossary 2025. en. <https://fuchsia.dev/fuchsia-src/glossary>. Visited 30 April 2025.

10 Evans, Pete. Jan. 2022. “Still Using a Classic BlackBerry? It’s Going to Stop Working Today, Company Says”. en-CA. In: CBC News. <https://www.cbc.ca/news/business/blackberry-network-1.6302558>. Visited 30 April 2025.

11 Bradshaw, Kyle. July 2018. “Fuchsia Friday: Respecting User Privacy after All.” en-US. <https://9to5google.com/2018/07/20/fuchsia-friday-respecting-user-privacy/>. Visited 24 April 2025.

12 Biggs, Simon; Lee, Damon & Heiser, Gernot. 2018. “The Jury Is in: Monolithic OS design Is Flawed”. In: *Asia-Pacific Workshop on Systems (APSys). Korea: ACM SIGOPS*.

13 Lameter, Christoph. 2007. “Extreme High Performance Computing or Why Microkernels Suck”. In: *Proceedings of the Linux Symposium*. Vol. 1, pp. 251–262.

14 Stankov, Ivan & Spasov, Grisha. 2006. “Discussion of Microkernel and Monolithic Kernel Approaches”. In: *International Scientific Conference Computer Science*. sn.

15 Suann, Jack. 2018. “Zircon on seL4”. The University of New South Wales.

16 Bradshaw, Kyle. June 2021. “How to Know if Your Nest Hub Has Been Updated to Fuchsia OS” Online. <https://9to5google.com/2021/06/11/how-to-know-fuchsia-os-nest-hub-update/>. Visited 8 October 2024.

17 Bradshaw, Kyle. Jan. 2023. “Google Accidentally Reveals an Upcoming Device Will Launch with Fuchsia” Online. <https://9to5google.com/2023/01/10/google- fuchsia-launch-upcoming-device/>. Visited 8 October 2024.

18 Amadeo, Ron. Jan. 2018. “Google’s Fuchsia OS on the Pixelbook: It Works! It Actually Works!” Online. <https://arstechnica.com/gadgets/2018/01/googles-fuchsia-os-on-the-pixelbook-it-works-it-actually-works/>. Visited 8 October 2024.

19 Amadeo, Ron. Dec. 2020. “Google’s Secretive Fuchsia OS is Open for Contributions” Online. <https://arstechnica.com/gadgets/2020/12/googles-secretive-fuchsia-os-is-open-for-contributions/>. Visited 8 October 2024.

20 Google. Sept. 2024a. “Get Started with Fuchsia” Online. <https://fuchsia.dev/fuchsia-src/get-started/get_fuchsia_source?hl=en>. Visited 10 September 2024.

21 Google. Sept. 2024b. “Get Started with Fuchsia” Online. <https://fuchsia.dev/fuchsia-src/get-started/get_fuchsia_source?hl=en#set-up-environment-variables>. Visited 10 September 2024.

22 Google. Sept. 2024c. “Fx Workflows” Online. <https://fuchsia.dev/fuchsia-src/development/build/fx#key-bundles>. Visited 10 September 2024.

23 Google. Dec. 2024d. “Build System” Online. <https://fuchsia.dev/fuchsia-src/get-started/learn/build/build-system>. Visited 17 December 2024.

24 ArchWiki. “Benchmarking” Online. <https://wiki.archlinux.org/title/Benchmarking>. Visited 28 April 2025.

25 GitHub. “Shellspec/shellbench” Mar. 2025. <https://github.com/shellspec/shellbench>. Visited 28 April 2025.

1 Full system information for Alpine containers

QEMU Alpine container system information

```
  test@alpine9 ~/shellbench (master)> inxi -Fxz
System:
  Kernel: 6.6.14-0-lts arch: x86_64 bits: 64 compiler: gcc v: 13.2.1
    Console: pty pts/0 Distro: Alpine Linux v3.19
Machine:
  Type: Qemu System: QEMU product: Standard PC (Q35 + ICH9, 2009)
    v: pc-q35-9.2 serial: <superuser required>
  Mobo: N/A model: N/A serial: N/A BIOS: SeaBIOS
    v: rel-1.16.3-2-gc13ff2cd-prebuilt.qemu.org date: 04/01/2014
CPU:
  Info: 4x 1-core model: AMD Ryzen 5 7600X bits: 64 type: SMP arch: Zen 4
    rev: 2 cache: L1: 4x 128 KiB (512 KiB) L2: 4x 512 KiB (2 MiB)
    L3: 4x 16 MiB (64 MiB)
  Speed (MHz): avg: 4691 min/max: N/A cores: 1: 4691 2: 4691 3: 4691 4: 4691
    bogomips: 37545
  Flags: avx avx2 lm nx pae sse sse2 sse3 sse4_1 sse4_2 sse4a ssse3 svm
Graphics:
  Message: Required tool lspci not installed. Check --recommends
  Display: server: No display server data found. Headless machine?
    tty: 80x40
  API: N/A Message: No API data available in console. Headless machine?
Audio:
  Message: No device data found.
  API: ALSA v: k6.6.14-0-lts status: kernel-api
Network:
  Message: Required tool lspci not installed. Check --recommends
  IF-ID-1: eth0 state: up speed: -1 duplex: unknown mac: <filter>
Drives:
  Local Storage: total: 4 GiB used: 0 KiB (0.0%)
  ID-1: /dev/vda model: N/A size: 4 GiB
Partition:
  Message: No partition data found.
Swap:
  Alert: No swap data was found.
Sensors:
  Src: lm-sensors Missing: Required tool sensors not installed. Check
    --recommends
Info:
  Processes: 162 Uptime: 4m Memory: total: 8 GiB available: 7.76 GiB
  used: 464.4 MiB (5.8%) Init: SysVinit rc: OpenRC v: 0.52.1 runlevel: default
  Compilers: N/A Packages: 62 Shell: ash (busybox) inxi: 3.3.31
```

Fuchsia Starnix full system information

```
/tmp/shellbench-master # inxi -Fxz
Use of uninitialized value $ENV{"HOME"} in concatenation (.) or string at /usr/
    bin/inxi line 637.
Use of uninitialized value in concatenation (.) or string at /usr/bin/inxi line
    642.
Use of uninitialized value $ENV{"HOME"} in concatenation (.) or string at /usr/
    bin/inxi line 648.
Use of uninitialized value in concatenation (.) or string at /usr/bin/inxi line
    653.
Use of uninitialized value $ENV{"HOME"} in concatenation (.) or string at /usr/
    bin/inxi line 670.
Use of uninitialized value $ENV{"HOME"} in concatenation (.) or string at /usr/
    bin/inxi line 687.
System:
  Kernel: 5.10.199-starnix arch: x86_64 bits: 64 compiler: clang v: HEAD
    Console: pty pts/0 Distro: Alpine Linux v3.19
Machine:
  Message: No machine data: try newer kernel. Is dmidecode installed? Try -M
    --dmidecode.
CPU:
  Info: quad core model: QEMU Virtual CPU bits: 64 type: MCP arch: N/A
    cache: N/A
  Speed: N/A min/max: N/A cores: No per core speed data found. bogomips: N/A
  Flags: N/A
Graphics:
  Message: Required tool lspci not installed. Check --recommends
  Display: server: No display server data found. Headless machine?
    tty: 80x40
  API: N/A Message: No API data available in console. Headless machine?
Audio:
  Message: No device data found.
Network:
  Message: Required tool lspci not installed. Check --recommends
  IF-ID-1: ethp0004 state: N/A speed: N/A duplex: N/A mac: N/A
  IF-ID-2: ifb0 state: N/A speed: N/A duplex: N/A mac: N/A
Drives:
  Local Storage: total: 0 KiB used: 0 KiB
Partition:
  Message: No partition data found.
Swap:
  Alert: No swap data was found.
Sensors:
  Src: lm-sensors Missing: Required tool sensors not installed. Check
    --recommends
Info:
  Processes: 6 Uptime: 5d 4h 44m Memory: total: N/A available: 8 GiB
```

used: 3.83 GiB (47.9%) Init: SysVinit Compilers: gcc: 13.2.1 Packages: 115
Shell: ash (busybox) inxi: 3.3.31

2 Tests failed by Starnix

The following tests were successful in QEMU and failed to run in the Starnix container.

2.1 QEMU Alpine container results

```
/home/test # cryptsetup benchmark
# Tests are approximate using memory only (no storage IO).
PBKDF2-sha1    2364320 iterations per second for 256-bit key
PBKDF2-sha256  4144569 iterations per second for 256-bit key
PBKDF2-sha512  2380422 iterations per second for 256-bit key
PBKDF2-ripemd160 1161213 iterations per second for 256-bit key
PBKDF2-whirlpool 923042 iterations per second for 256-bit key
argon2i       15 iterations, 1048576 memory, 4 parallel threads (CPUs) for 256-bit
    key (requested 2000 ms time)
argon2id      14 iterations, 1048576 memory, 4 parallel threads (CPUs) for 256-bit
    key (requested 2000 ms time)
#      Algorithm |       Key |      Encryption |      Decryption
           aes-cbc       128b       334.5 MiB/s       386.7 MiB/s
       serpent-cbc       128b       156.4 MiB/s      1061.0 MiB/s
       twofish-cbc       128b       309.9 MiB/s       686.5 MiB/s
           aes-cbc       256b       277.4 MiB/s       302.6 MiB/s
       serpent-cbc       256b       164.3 MiB/s      1042.0 MiB/s
       twofish-cbc       256b       318.8 MiB/s       684.6 MiB/s
           aes-xts       256b      3884.6 MiB/s      4354.5 MiB/s
       serpent-xts       256b       926.6 MiB/s       948.3 MiB/s
       twofish-xts       256b       605.9 MiB/s       639.4 MiB/s
           aes-xts       512b      4007.5 MiB/s      4019.1 MiB/s
       serpent-xts       512b       956.2 MiB/s       944.7 MiB/s
       twofish-xts       512b       625.6 MiB/s       635.5 MiB/s


/home/test # sysbench fileio prepare
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

128 files, 16384Kb each, 2048Mb total
Creating files for the test...
Extra file open flags: (none)
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2
Creating file test_file.3
Creating file test_file.4
```

```
Creating file test_file.5
Creating file test_file.6
Creating file test_file.7
Creating file test_file.8
Creating file test_file.9
Creating file test_file.10
Creating file test_file.11
Creating file test_file.12
Creating file test_file.13
Creating file test_file.14
Creating file test_file.15
Creating file test_file.16
Creating file test_file.17
Creating file test_file.18
Creating file test_file.19
Creating file test_file.20
Creating file test_file.21
Creating file test_file.22
Creating file test_file.23
Creating file test_file.24
Creating file test_file.25
Creating file test_file.26
Creating file test_file.27
Creating file test_file.28
Creating file test_file.29
Creating file test_file.30
Creating file test_file.31
Creating file test_file.32
Creating file test_file.33
Creating file test_file.34
Creating file test_file.35
Creating file test_file.36
Creating file test_file.37
Creating file test_file.38
Creating file test_file.39
Creating file test_file.40
Creating file test_file.41
Creating file test_file.42
Creating file test_file.43
Creating file test_file.44
Creating file test_file.45
Creating file test_file.46
Creating file test_file.47
Creating file test_file.48
Creating file test_file.49
Creating file test_file.50
Creating file test_file.51
```

```
Creating file test_file.52
Creating file test_file.53
Creating file test_file.54
Creating file test_file.55
Creating file test_file.56
Creating file test_file.57
Creating file test_file.58
Creating file test_file.59
Creating file test_file.60
Creating file test_file.61
Creating file test_file.62
Creating file test_file.63
Creating file test_file.64
Creating file test_file.65
Creating file test_file.66
Creating file test_file.67
Creating file test_file.68
Creating file test_file.69
Creating file test_file.70
Creating file test_file.71
Creating file test_file.72
Creating file test_file.73
Creating file test_file.74
Creating file test_file.75
Creating file test_file.76
Creating file test_file.77
Creating file test_file.78
Creating file test_file.79
Creating file test_file.80
Creating file test_file.81
Creating file test_file.82
Creating file test_file.83
Creating file test_file.84
Creating file test_file.85
Creating file test_file.86
Creating file test_file.87
Creating file test_file.88
Creating file test_file.89
Creating file test_file.90
Creating file test_file.91
Creating file test_file.92
Creating file test_file.93
Creating file test_file.94
Creating file test_file.95
Creating file test_file.96
Creating file test_file.97
Creating file test_file.98
```

```
Creating file test_file.99
Creating file test_file.100
Creating file test_file.101
Creating file test_file.102
Creating file test_file.103
Creating file test_file.104
Creating file test_file.105
Creating file test_file.106
Creating file test_file.107
Creating file test_file.108
Creating file test_file.109
Creating file test_file.110
Creating file test_file.111
Creating file test_file.112
Creating file test_file.113
Creating file test_file.114
Creating file test_file.115
Creating file test_file.116
Creating file test_file.117
Creating file test_file.118
Creating file test_file.119
Creating file test_file.120
Creating file test_file.121
Creating file test_file.122
Creating file test_file.123
Creating file test_file.124
Creating file test_file.125
Creating file test_file.126
Creating file test_file.127
2147483648 bytes written in 0.59 seconds (3444.91 MiB/sec).

/home/test # sysbench fileio --file-test-mode=rndrw run
sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time


Extra file open flags: (none)
128 files, 16MiB each
2GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
```

```
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!


File operations:
    reads/s:                      263265.03
    writes/s:                     175510.02
    fsyncs/s:                     561640.39

Throughput:
    read, MiB/s:                  4113.52
    written, MiB/s:               2742.34

General statistics:
    total time:                          10.0001s
    total number of events:              10004675

Latency (ms):
         min:                                    0.00
         avg:                                    0.00
         max:                                    0.36
         95th percentile:                        0.00
         sum:                                 9164.27

Threads fairness:
    events (avg/stddev):           10004675.0000/0.00
    execution time (avg/stddev): 9.1643/0.00

    /tmp # sysbench fileio prepare
    sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

    128 files, 16384Kb each, 2048Mb total
    Creating files for the test...
    Extra file open flags: (none)
    Creating file test_file.0
    Creating file test_file.1
    [......]
    Creating file test_file.127
    2147483648 bytes written in 4.77 seconds (429.25 MiB/sec).
    /tmp # sysbench fileio --file-test-mode=rndrw run
    sysbench 1.0.20 (using bundled LuaJIT 2.1.0-beta2)

    Running the test with following options:
    Number of threads: 1
```

```
  Initializing random number generator from current time


  Extra file open flags: (none)
  128 files, 16MiB each
  2GiB total file size
  Block size 16KiB
  Number of IO requests: 0
  Read/Write ratio for combined random IO test: 1.50
  Periodic FSYNC enabled, calling fsync() each 100 requests.
  Calling fsync() at the end of test, Enabled.
  Using synchronous I/O mode
  Doing random r/w test
  Initializing worker threads...

  FATAL: sb_thread_create() for thread #0 failed. errno = 0 (No error
      information)

/home/test # date; phoronix-test-suite benchmark fio; date
Wed Apr 30 07:58:20 EEST 2025

Updated OpenBenchmarking.org Repository Index
pts: 557 Distinct Tests, 2529 Test Versions, 57 Suites
Available Changes From 3 July 2022 To 30 April
New Test:       pts/3dmark              v1.0.0 3DMark Wild Life Extreme
Updated Test:   pts/ai-benchmark        v1.0.2 AI Benchmark Alpha
Updated Test:   pts/aircrack-ng         v1.3.0 Aircrack-ng
Updated Test:   pts/aom-av1             v3.12.0 AOM AV1
Updated Test:   pts/apache              v3.0.0 Apache HTTP Server
New Test:       pts/apache-iotdb        v1.2.0 Apache IoTDB
Updated Test:   pts/apache-siege        v1.1.0 Apache Siege
Updated Test:   pts/arrayfire           v1.2.1 ArrayFire
Updated Test:   pts/astcenc             v1.6.0 ASTC Encoder
Updated Test:   pts/avifenc             v1.4.1 libavif avifenc
New Test:       pts/axs2mlperf          v1.0.0 axs2mlperf
Updated Test:   pts/basis               v1.2.0 Basis Universal
Updated Test:   pts/batman-knight       v1.0.1 Batman: Arkham Knight
Updated Test:   pts/blender             v4.4.0 Blender
Updated Test:   pts/blosc               v1.3.0 C-Blosc
New Test:       pts/breaking-limit      v1.0.1 GPUScore: Breaking Limit
Updated Test:   pts/brl-cad             v1.6.0 BRL-CAD
Updated Test:   pts/build-eigen         v1.2.0 Timed Eigen Compilation
Updated Test:   pts/build-erlang        v1.2.0 Timed Erlang/OTP Compilation
Updated Test:   pts/build-ffmpeg        v7.0.0 Timed FFmpeg Compilation
Updated Test:   pts/build-gcc           v1.4.0 Timed GCC Compilation
Updated Test:   pts/build-gem5          v1.1.0 Timed Gem5 Compilation
Updated Test:   pts/build-godot         v4.0.0 Timed Godot Game Engine Compilation
```

Updated Test: pts/build-linux-kernel v1.16.0 Timed Linux Kernel Compilation
Updated Test: pts/build-llvm v1.5.0 Timed LLVM Compilation
Updated Test: pts/build-mesa v1.1.0 Timed Mesa Compilation
Updated Test: pts/build-nodejs v1.4.0 Timed Node.js Compilation
Updated Test: pts/build-php v1.7.0 Timed PHP Compilation
New Test: pts/build-python v1.0.0 Timed CPython Compilation
Updated Test: pts/build-wasmer v1.2.0 Timed Wasmer Compilation
Updated Test: pts/build2 v1.3.0 Build2
Updated Test: pts/byte v1.3.0 BYTE Unix Benchmark
Updated Test: pts/c-ray v2.0.0 C-Ray
Updated Test: pts/cachebench v1.2.0 CacheBench
Updated Test: pts/cassandra v1.3.1 Apache Cassandra
Updated Test: pts/clickhouse v1.2.0 ClickHouse
Updated Test: pts/cloudsuite-da v1.2.0 CloudSuite Data Analytics
Updated Test: pts/cloudsuite-ga v1.1.0 CloudSuite Graph Analytics
Updated Test: pts/cloudsuite-ma v1.1.0 CloudSuite In-Memory Analytics
Updated Test: pts/cloudsuite-ms v1.0.1 CloudSuite Media Streaming
Updated Test: pts/cloudsuite-ws v1.1.0 CloudSuite Web Serving
Updated Test: pts/cloverleaf v1.2.0 CloverLeaf
Updated Test: pts/clpeak v1.1.0 clpeak
New Test: pts/cockroach v1.0.2 CockroachDB
Updated Test: pts/compress-7zip v1.11.0 7-Zip Compression
Updated Test: pts/compress-lz4 v1.10.0 LZ4 Compression
Updated Test: pts/compress-pbzip2 v1.6.1 Parallel BZIP2 Compression
Updated Test: pts/compress-zstd v1.6.0 Zstd Compression
Updated Test: pts/couchdb v1.4.1 Apache CouchDB
Updated Test: pts/cp2k v1.5.0 CP2K Molecular Dynamics
Updated Test: pts/cpuminer-opt v1.8.0 Cpuminer-Opt
Updated Test: pts/cryptopp v1.1.0 Crypto++
New Test: pts/cs2 v1.0.2 Counter-Strike 2
Updated Test: pts/csgo v1.7.2 Counter-Strike: Global Offensive
New Test: pts/cyberpunk2077 v1.0.1 Cyberpunk 2077
Updated Test: pts/dacapobench v1.1.0 DaCapo Benchmark
Updated Test: pts/daphne v1.1.0 Darmstadt Automotive Parallel Heterogeneous Suite
Updated Test: pts/dav1d v1.16.0 dav1d
Updated Test: pts/ddnet v1.4.1 DDraceNetwork
New Test: pts/deeprec v1.0.2 DeepRec
New Test: pts/deepsparse v1.7.0 Neural Magic DeepSparse
Updated Test: pts/draco v1.6.1 Google Draco
New Test: pts/dragonflydb v1.1.0 Dragonflydb
New Test: pts/duckdb v1.1.0 DuckDB
New Test: pts/easywave v1.0.0 easyWave
Updated Test: pts/embree v1.6.1 Embree
Updated Test: pts/encode-flac v1.9.0 FLAC Audio Encoding
Updated Test: pts/encode-opus v1.5.0 Opus Codec Encoding
Updated Test: pts/encode-wavpack v1.5.0 WavPack Audio Encoding

New Test: pts/encodec v1.0.1 EnCodec
New Test: pts/epoch v1.0.0 Epoch
Updated Test: pts/espeak v1.7.0 eSpeak-NG Speech Engine
New Test: pts/etcd v1.0.0 etcd
Updated Test: pts/etcpak v2.0.0 Etcpak
Updated Test: pts/ethr v1.2.1 Ethr
Updated Test: pts/etlegacy v1.3.0 ET: Legacy
Updated Test: pts/f12021 v1.0.1 F1 2021
New Test: pts/f122 v1.0.3 F1 22
New Test: pts/faiss v1.0.1 Faiss
Updated Test: pts/ffmpeg v7.0.1 FFmpeg
Updated Test: pts/fio v2.1.0 Flexible IO Tester
New Test: pts/fluidx3d v1.5.0 FluidX3D
New Test: pts/furmark v1.0.0 FurMark
Updated Test: pts/gcrypt v1.2.0 Gcrypt Library
Updated Test: pts/geekbench v6.1.0 Geekbench
Updated Test: pts/glibc-bench v1.9.0 Glibc Benchmarks
Updated Test: pts/glmark2 v1.4.0 GLmark2
Updated Test: pts/gpaw v1.2.0 GPAW
New Test: pts/gpuowl v1.1.0 GpuOwl
Updated Test: pts/graph500 v1.0.2 Graph500
Updated Test: pts/graphics-magick v2.2.0 GraphicsMagick
Updated Test: pts/gravitymark v1.10.0 GravityMark
Updated Test: pts/gromacs v1.10.0 GROMACS
New Test: pts/gta5 v1.0.0 Grand Theft Auto V
New Test: pts/hadoop v1.0.0 Apache Hadoop
Updated Test: pts/hammerdb-mariadb v1.1.0 HammerDB - MariaDB
Updated Test: pts/hammerdb-postgresql v1.1.1 HammerDB - PostgreSQL
Updated Test: pts/hbase v1.1.0 Apache HBase
New Test: pts/heffte v1.1.0 HeFFTe - Highly Efficient FFT for Exascale
Updated Test: pts/hitman v1.0.2 HITMAN
Updated Test: pts/hitman3 v1.0.1 HITMAN 3
Updated Test: pts/hpcg v1.3.0 High Performance Conjugate Gradient
Updated Test: pts/influxdb v1.0.2 InfluxDB
Updated Test: pts/intel-mlc v1.1.0 Intel Memory Latency Checker
New Test: pts/intel-tensorflow v1.0.0 Intel TensorFlow
Updated Test: pts/iperf v1.2.0 iPerf
Updated Test: pts/java-scimark2 v1.2.0 Java SciMark
Updated Test: pts/john-the-ripper v1.8.0 John The Ripper
Updated Test: pts/jpegxl v1.6.0 JPEG-XL libjxl
Updated Test: pts/jpegxl-decode v1.6.0 JPEG-XL Decoding libjxl
Updated Test: pts/keydb v1.4.0 KeyDB
Updated Test: pts/kripke v1.2.0 Kripke
Updated Test: pts/kvazaar v1.2.0 Kvazaar
Updated Test: pts/l4d2 v1.0.1 Left 4 Dead 2
New Test: pts/laghos v1.0.1 Laghos

Updated Test: pts/lammps v1.4.0 LAMMPS Molecular Dynamics Simulator
Updated Test: pts/lczero v1.8.0 LeelaChessZero
Updated Test: pts/leveldb v1.1.0 LevelDB
Updated Test: pts/libplacebo v1.2.0 Libplacebo
New Test: pts/libxsmm v1.0.1 libxsmm
Updated Test: pts/liquid-dsp v1.7.0 Liquid-DSP
New Test: pts/litert v1.0.0 LiteRT
New Test: pts/llama-cpp v2.1.1 Llama.cpp
New Test: pts/llamafile v1.3.0 Llamafile
Updated Test: pts/lzbench v1.2.0 lzbench
New Test: pts/mariadb v1.1.0 MariaDB
New Test: pts/memcached v1.2.0 Memcached
Updated Test: pts/memtier-benchmark v1.5.0 Redis 7.0.12 + memtier_benchmark
New Test: pts/minibude v1.0.0 miniBUDE
Updated Test: pts/mnn v3.1.0 Mobile Neural Network
Updated Test: pts/mocassin v1.1.0 Monte Carlo Simulations of Ionised Nebulae
Updated Test: pts/mt-dgemm v1.3.1 ACES DGEMM
Updated Test: pts/mysqlslap v1.5.0 MariaDB mariadb-slap
Updated Test: pts/namd v1.4.0 NAMD
Updated Test: pts/namd-cuda v1.2.0 NAMD CUDA
Updated Test: pts/natron v1.1.0 Natron
Updated Test: pts/ncnn v1.6.0 NCNN
Updated Test: pts/neatbench v1.1.0 NeatBench
New Test: pts/nekrs v1.1.0 nekRS
Updated Test: pts/nginx v3.0.1 nginx
Updated Test: pts/node-web-tooling v1.0.1 Node.js V8 Web Tooling Benchmark
Updated Test: pts/numenta-nab v1.1.1 Numenta Anomaly Benchmark
Updated Test: pts/nwchem v1.2.0 NWChem
Updated Test: pts/oidn v2.3.0 Intel Open Image Denoise
Updated Test: pts/onednn v3.6.0 oneDNN
Updated Test: pts/onnx v1.19.0 ONNX Runtime
New Test: pts/opencl-benchmark v1.1.0 ProjectPhysX OpenCL-Benchmark
Updated Test: pts/opencv v1.3.0 OpenCV
New Test: pts/openems v1.0.0 OpenEMS
Updated Test: pts/openfoam v1.2.0 OpenFOAM
New Test: pts/openradioss v1.2.0 OpenRadioss
Updated Test: pts/openssl v3.3.0 OpenSSL
Updated Test: pts/openvino v1.6.0 OpenVINO
New Test: pts/openvino-genai v1.0.0 OpenVINO GenAI
Updated Test: pts/openvkl v2.0.0 OpenVKL
Updated Test: pts/ospray v3.2.0 OSPRay
Updated Test: pts/ospray-studio v1.3.0 OSPRay Studio
New Test: pts/palabos v1.0.1 Palabos
Updated Test: pts/paraview v1.4.1 ParaView
Updated Test: pts/perf-bench v1.1.0 perf-bench
New Test: pts/petsc v1.0.0 PETSc

Updated Test:	pts/pgbench	v1.15.0 PostgreSQL
Updated Test:	pts/portal2	v1.1.2 Portal 2
Updated Test:	pts/primesieve	v1.12.0 Primesieve
Updated Test:	pts/pyperformance	v1.1.0 PyPerformance
New Test:	pts/pytorch	v1.2.0 PyTorch
Updated Test:	pts/qmcpack	v1.8.0 QMCPACK
New Test:	pts/quadray	v1.0.0 QuadRay
Updated Test:	pts/quake2rtx	v1.8.0 Quake II RTX
Updated Test:	pts/quantlib	v2.1.0 QuantLib
New Test:	pts/quicksilver	v1.0.0 Quicksilver
New Test:	pts/rabbitmq	v1.1.1 RabbitMQ
Updated Test:	pts/rav1e	v1.8.0 rav1e
Updated Test:	pts/redis	v1.4.0 Redis
Updated Test:	pts/relion	v1.2.0 RELION
New Test:	pts/remhos	v1.0.0 Remhos
Updated Test:	pts/renaissance	v1.4.0 Renaissance
Updated Test:	pts/rnnoise	v1.1.0 RNNoise
Updated Test:	pts/rocksdb	v1.7.0 RocksDB
Updated Test:	pts/rtiv	v1.1.0 Ray Tracing In Vulkan
New Test:	pts/rustls	v1.0.0 Rustls
Updated Test:	pts/schbench	v1.2.0 Schbench
Updated Test:	pts/scikit-learn	v2.0.0 Scikit-Learn
New Test:	pts/scylladb	v1.0.0 ScyllaDB
Updated Test:	pts/simdjson	v2.1.0 simdjson
Updated Test:	pts/smhasher	v1.1.0 SMHasher
New Test:	pts/spacy	v1.0.0 spaCy
New Test:	pts/spark	v1.0.1 Apache Spark
New Test:	pts/spark-tpcds	v1.0.0 Apache Spark TPC-DS
New Test:	pts/spark-tpch	v1.0.0 Apache Spark TPC-H
Updated Test:	pts/spec-jbb2015	v1.1.0 SPECjbb 2015
New Test:	pts/specfem3d	v1.1.0 SPECFEM3D
New Test:	pts/speedb	v1.0.1 Speedb
Updated Test:	pts/sqlite	v2.2.0 SQLite
Updated Test:	pts/srsran	v2.4.1 srsRAN Project
New Test:	pts/star-swarm	v1.0.0 Star Swarm Stress Test
Updated Test:	pts/stargate	v1.1.0 Stargate Digital Audio Workstation
Updated Test:	pts/stockfish	v1.7.0 Stockfish
Updated Test:	pts/strange-brigade	v1.0.1 Strange Brigade
Updated Test:	pts/stream	v1.3.4 Stream
Updated Test:	pts/stress-ng	v1.13.0 Stress-NG
Updated Test:	pts/supertuxkart	v1.7.1 SuperTuxKart
Updated Test:	pts/svt-av1	v2.16.0 SVT-AV1
Updated Test:	pts/tensorflow	v2.2.0 TensorFlow
Updated Test:	pts/tf2	v1.2.4 Team Fortress 2
New Test:	pts/tidb	v1.0.3 TiDB Community Server
Updated Test:	pts/tww3	v1.0.1 Total War: WARHAMMER III
Updated Test:	pts/unigine-heaven	v1.6.6 Unigine Heaven

Updated Test: pts/unigine-super v1.0.8 Unigine Superposition
Updated Test: pts/unigine-valley v1.1.9 Unigine Valley
Updated Test: pts/unpack-linux v1.2.0 Unpacking The Linux Kernel
Updated Test: pts/unvanquished v1.9.0 Unvanquished
New Test: pts/uvg266 v1.1.0 uvg266
Updated Test: pts/v-ray v1.5.0 Chaos Group V-RAY
New Test: pts/valkey v1.0.0 Valkey
Updated Test: pts/vkfft v1.3.0 VkFFT
Updated Test: pts/vkmark v1.3.2 VKMark
Updated Test: pts/vkpeak v1.2.0 vkpeak
Updated Test: pts/vkresample v1.0.2 VkResample
Updated Test: pts/vpxenc v3.2.0 VP9 libvpx Encoding
New Test: pts/vvenc v1.13.0 VVenC
New Test: pts/warpx v1.0.0 WarpX
Updated Test: pts/webp v1.4.0 WebP Image Encode
Updated Test: pts/webp2 v1.2.1 WebP2 Image Encode
New Test: pts/whisper-cpp v1.1.0 Whisper.cpp
New Test: pts/whisperfile v1.0.0 Whisperfile
New Test: pts/will-it-scale v1.0.0 will-it-scale
Updated Test: pts/x265 v1.5.0 x265
Updated Test: pts/xmrig v1.2.0 Xmrig
New Test: pts/xnnpack v1.1.0 XNNPACK
Updated Test: pts/xonotic v1.7.0 Xonotic
New Test: pts/xplane12 v1.0.1 X-Plane
Updated Test: pts/y-cruncher v1.5.0 Y-Cruncher
New Test: pts/yugabytedb v1.0.0 YugabyteDB
New Test: pts/z3 v1.0.1 Z3 Theorem Prover
New Test: pts/zendnn-tensorflow v1.0.0 AMD ZenDNN TensorFlow
Updated Suite: pts/compilation v1.2.8 Timed Code Compilation
Updated Suite: pts/database v1.3.13 Database Test Suite
Updated Suite: pts/electronic-design v1.0.1 Electronic Design
Updated Suite: pts/hpc v1.1.12 HPC - High Performance Computing
New Suite: pts/llm v1.0.0 Large Language Models
Updated Suite: pts/machine-learning v1.3.13 Machine Learning
Updated Suite: pts/opencl v1.1.1 OpenCL
Updated Suite: pts/raytracing v1.0.3 Raytracing
Updated Suite: pts/server v1.4.2 Server
Updated Suite: pts/steam v1.0.10 Steam
Updated Suite: pts/video-encoding v1.3.4 Video Encoding
Updated OpenBenchmarking.org Repository Index
system: 48 Distinct Tests, 148 Test Versions
Available Changes From 3 July 2022 To 30 April
New Test: system/cephfs-rados v1.0.1 CephFS RADOS Benchmark
New Test: system/compress-7zip v1.0.0 7-Zip Compression
New Test: system/graphics-magick v1.0.0 GraphicsMagick
New Test: system/gromacs v1.0.0 GROMACS
Updated Test: system/inkscape v1.0.1 Inkscape

```
Updated Test: system/openssl      v1.2.1 OpenSSL
New Test:     system/povray       v1.0.0 POV-Ray
Updated Test: system/selenium     v1.0.45 Selenium
New Test:     system/stockfish    v1.0.0 Stockfish
New Test:     system/x265         v1.0.0 x265
Updated OpenBenchmarking.org Repository Index
git: 9 Distinct Tests, 12 Test Versions
Available Changes From 3 July 2022 To 30 April
New Test: git/clickhouse v1.0.0 ClickHouse
    Evaluating External Test Dependencies
        --------------------------------------------------------------------------------


The following dependencies are needed and will be installed:

- gcc
- make
- build-base
- automake
- autoconf
- linux-headers
- libaio
- unzip
- bash

This process may take several minutes.
Please enter your root password below:
(1/28) Upgrading musl (1.2.4_git20230717-r4 -> 1.2.4_git20230717-r5)
(2/28) Installing m4 (1.4.19-r3)
(3/28) Installing perl (5.38.3-r1)
(4/28) Installing autoconf (2.71-r2)
(5/28) Installing automake (1.16.5-r2)
(6/28) Installing bash (5.2.21-r0)
Executing bash-5.2.21-r0.post-install
(7/28) Installing jansson (2.14-r4)
(8/28) Installing binutils (2.41-r1)
(9/28) Installing libmagic (5.45-r1)
(10/28) Installing file (5.45-r1)
(11/28) Installing libgomp (13.2.1_git20231014-r0)
(12/28) Installing libatomic (13.2.1_git20231014-r0)
(13/28) Installing gmp (6.3.0-r0)
(14/28) Installing isl26 (0.26-r1)
(15/28) Installing mpfr4 (4.2.1-r0)
(16/28) Installing mpc1 (1.3.1-r1)
(17/28) Installing gcc (13.2.1_git20231014-r0)
(18/28) Installing libstdc++-dev (13.2.1_git20231014-r0)
(19/28) Installing musl-dev (1.2.4_git20230717-r5)
```

```
(20/28) Installing libc-dev (0.7.2-r5)
(21/28) Installing g++ (13.2.1_git20231014-r0)
(22/28) Installing make (4.4.1-r2)
(23/28) Installing fortify-headers (1.1-r3)
(24/28) Installing patch (2.7.6-r10)
(25/28) Installing build-base (0.5-r3)
(26/28) Installing libaio (0.3.113-r2)
(27/28) Installing linux-headers (6.5-r0)
(28/28) Installing unzip (6.0-r14)
Executing busybox-1.36.1-r15.trigger
OK: 95 packages, 1986 dirs, 13810 files, 357 MiB

There are dependencies still missing from the system:
- Linux AIO

1: Ignore missing dependencies and proceed with installation.
2: Skip installing the tests with missing dependencies.
3: Re-attempt to install the missing dependencies.
4: Quit the current Phoronix Test Suite process.
Missing dependencies action: 3

The following dependencies are needed and will be installed:

- libaio

This process may take several minutes.
Please enter your root password below:
OK: 95 packages, 1986 dirs, 13810 files, 357 MiB

Phoronix Test Suite v10.8.4

    To Install:  pts/fio-2.1.0

    Determining File Requirements
        ----------------------------------------------------------------------

    Searching Download Caches
        ----------------------------------------------------------------------


    1 Test To Install
        1 File To Download [7.07MB]
        14MB Of Disk Space Is Needed
        22 Seconds Estimated Install Time

    pts/fio-2.1.0:
        Test Installation 1 of 1
```

1 File Needed [7.07 MB]
Downloading: fio-3.36.tar.gz

[7.07MB]
Downloading

Approximate Install Size: 14 MB
Estimated Install Time: 22 Seconds
Installing Test @ 04:58:44

Flexible IO Tester 3.36:
pts/fio-2.1.0
Disk Test Configuration
1: Random Read
2: Random Write
3: Sequential Read
4: Sequential Write
5: Test All Options
** Multiple items can be selected, delimit by a comma. **
Type: 1,2

1: IO_uring
2: POSIX AIO
3: Sync
4: Linux AIO
5: Test All Options
** Multiple items can be selected, delimit by a comma. **
Engine: 1

1: No
2: Yes
3: Test All Options
** Multiple items can be selected, delimit by a comma. **
Direct: 1

1: 4KB
2: 8KB
3: 16KB
4: 32KB
5: 64KB
6: 128KB

7: 256KB
8: 512KB
9: 1MB
10: 2MB
11: 4MB
12: 8MB
13: Test All Options
** Multiple items can be selected, delimit by a comma. **
Block Size: 1

1: 1
2: 2
3: 4
4: Test All Options
** Multiple items can be selected, delimit by a comma. **
Job Count: 1

System Information

[8192] str_split(): Passing null to parameter #1 ($string) of type string is deprecated in pts_strings:502

PROCESSOR:	4 x AMD Ryzen 5 7600X 6-Core
Core Count:	4
Extensions:	SSE 4.2 + AVX512_VNNI + AVX512CD + AVX2 + AVX + RDRAND + FSGSBASE
Cache Size:	0.5 MB
Microcode:	0xa601209
Core Family:	Family 25 Model 97
GRAPHICS:	qxldrmfb
Screen:	1024x768
MOTHERBOARD:	QEMU Standard PC
BIOS Version:	rel-1.16.3-2-gc13ff2cd-prebuilt.qemu.org
Audio:	QEMU Generic
MEMORY:	8GB

DISK: 4GB
File-System: tmpfs

OPERATING SYSTEM: Alpine Linux v3.19 3.19.1
Kernel: 6.6.14-0-lts (x86_64)
Compiler: GCC 13.2.1 20231014
Security: gather_data_sampling: Not affected
+ itlb_multihit: Not affected
+ l1tf: Not affected
+ mds: Not affected
+ meltdown: Not affected
+ mmio_stale_data: Not affected
+ retbleed: Not affected
+ spec_rstack_overflow: Mitigation of Safe RET
+ spec_store_bypass: Mitigation of SSB disabled via prctl
+ spectre_v1: Mitigation of usercopy/swapgs barriers and __user pointer sanitization
+ spectre_v2: Mitigation of Enhanced / Automatic IBRS IBPB: conditional STIBP: disabled RSB filling PBRSB-eIBRS: Not affected
+ srbds: Not affected
+ tsx_async_abort: Not affected

Would you like to save these test results (Y/n):
Enter a name for the result file: alpine-qemu-test1
Enter a unique name to describe this test run / configuration: alpine-qemu-test1

If desired, enter a new description below to better describe this result set / system configuration under test.
Press ENTER to proceed without changes.

Current Description: 4 x AMD Ryzen 5 7600X 6-Core testing with a QEMU Standard PC (Q35 + ICH9 2009) (rel-1.16.3-2-gc13ff2cd-prebuilt.qemu.org BIOS) and qxldrmfb on Alpine Linux v3.19 3.19.1 via the Phoronix Test Suite.

New Description:

[8192] str_split(): Passing null to parameter #1 ($string) of type string is deprecated in pts_strings:502

[8192] str_split(): Passing null to parameter #1 ($string) of type string is deprecated in pts_strings:502

Flexible IO Tester 3.36:
pts/fio-2.1.0 [Type: Random Read - Engine: IO_uring - Direct: No - Block Size: 4KB - Job Count: 1 - Disk Target: Default Test Directory]

Test 1 of 2
Estimated Trial Run Count: 3
Estimated Test Run-Time: 9 Minutes
Estimated Time To Completion: 17 Minutes [05:15 UTC]
Running Pre-Test Script @ 04:59:39
Started Run 1 @ 04:59:39
Running Interim Test Script @ 05:01:07
Started Run 2 @ 05:01:09
Running Interim Test Script @ 05:02:37
Started Run 3 @ 05:02:39
Running Interim Test Script @ 05:04:07
Started Run 4 @ 05:04:09 *
Running Interim Test Script @ 05:05:36
Started Run 5 @ 05:05:38 *
Running Interim Test Script @ 05:07:06
Started Run 6 @ 05:07:08 *
Running Interim Test Script @ 05:08:36
Started Run 7 @ 05:08:38 *
Running Interim Test Script @ 05:10:05
Started Run 8 @ 05:10:07 *
Running Interim Test Script @ 05:11:35
Started Run 9 @ 05:11:37 *
Running Interim Test Script @ 05:13:05
Started Run 10 @ 05:13:07 *
Running Interim Test Script @ 05:14:34
Started Run 11 @ 05:14:37 *
Running Interim Test Script @ 05:16:04
Started Run 12 @ 05:16:06 *
Running Interim Test Script @ 05:17:34
Started Run 13 @ 05:17:36 *
Running Interim Test Script @ 05:19:04
Started Run 14 @ 05:19:06 *
Running Interim Test Script @ 05:20:33
Started Run 15 @ 05:20:35 *
Running Post-Test Script @ 05:22:03

Type: Random Read - Engine: IO_uring - Direct: No - Block Size: 4KB - Job Count: 1 - Disk Target: Default Test Directory:
4470
4514
4062
4519
4516
4499
4181
4503
4562

4487
4546
4475
4470
4408
4408

Average: 4441 MB/s
Deviation: 3.12%
Samples: 15

Comparison of 45 OpenBenchmarking.org samples since 2 December 2023; median result: 503 MB/s. Box plot of samples:
[---############!##*########* ---------- |
*]

^ 2000GB Samsung SSD 980 PRO 2TB: 889
^ 2048GB SOLIDIGM SSDPFKKW020X7: 608

Type: Random Read - Engine: IO_uring - Direct: No - Block Size: 4KB - Job Count: 1 - Disk Target: Default Test Directory:
1144000
1156000
1040000
1157000
1156000
1152000
1070000
1153000
1168000
1149000
1164000
1146000
1144000
1129000
1128000

Average: 1137067 IOPS

Deviation: 3.12%
Samples: 15

Comparison of 48 OpenBenchmarking.org samples since 2 December 2023; median result: 156333 IOPS. Box plot of samples:
[-----#############!########*##### ---------- |
*]

^ 2000GB Samsung SSD 980 PRO 2TB: 227604

Flexible IO Tester 3.36:
pts/fio-2.1.0 [Type: Random Write - Engine: IO_uring - Direct: No - Block Size: 4KB - Job Count: 1 - Disk Target: Default Test Directory]
Test 2 of 2
Estimated Trial Run Count: 3
Estimated Time To Completion: 5 Minutes [05:26 UTC]
Running Pre-Test Script @ 05:22:13
Started Run 1 @ 05:22:13
Running Interim Test Script @ 05:23:40
Started Run 2 @ 05:23:42
Running Interim Test Script @ 05:25:09
Started Run 3 @ 05:25:12
Running Interim Test Script @ 05:26:39
Started Run 4 @ 05:26:41 *
Running Post-Test Script @ 05:28:08

Type: Random Write - Engine: IO_uring - Direct: No - Block Size: 4KB - Job Count: 1 - Disk Target: Default Test Directory:
3976
4092
4200
4070

Average: 4085 MB/s
Deviation: 2.25%
Samples: 4

Comparison of 26 OpenBenchmarking.org samples since 2 December 2023; median result: 692 MB/s. Box plot of samples:

[
|-----##############!##############-----------------------------*----------------
*]

Western Digital WD_BLACK SN850X: 2023 ^

Type: Random Write - Engine: IO_uring - Direct: No - Block Size: 4KB - Job Count: 1 - Disk Target: Default Test Directory:
1018000
1047000
1075000
1042000

Average: 1045500 IOPS
Deviation: 2.24%
Samples: 4

Comparison of 29 OpenBenchmarking.org samples since 2 December 2023; median result: 178333 IOPS. Box plot of samples:
[
|----------########!##############################-----------------*----------------
*]

Western Digital WD_BLACK SN850X: 517889 ^

Do you want to view the text results of the testing (Y/n): n
Would you like to upload the results to OpenBenchmarking.org (y/n): n

2.2 Fuchsia Starnix Alpine container

```
/tmp # cryptsetup benchmark
# Tests are approximate using memory only (no storage IO).
PBKDF2-sha1   2191381 iterations per second for 256-bit key
PBKDF2-sha256  3584875 iterations per second for 256-bit key
PBKDF2-sha512  1762312 iterations per second for 256-bit key
PBKDF2-ripemd160 1073261 iterations per second for 256-bit key
PBKDF2-whirlpool 821768 iterations per second for 256-bit key
argon2i     N/A
argon2id    N/A
Required kernel crypto interface not available.
Ensure you have algif_skcipher kernel module loaded.

Phoronix Test Suite v10.8.4

    To Install:  pts/fio-2.1.0

    Determining File Requirements
        ----------------------------------------------------
    Searching Download Caches
        ----------------------------------------------------------

    1 Test To Install
        1 File To Download [7.07MB]
        14MB Of Disk Space Is Needed
        22 Seconds Estimated Install Time

    pts/fio-2.1.0:
        Test Installation 1 of 1
        1 File Needed [7.07 MB]
        Downloading:    fio-3.36.tar.gz

            [7.07MB]
        Downloading
            ----------------------------------------------------------------

        Approximate Install Size: 14 MB
        Estimated Install Time: 22 Seconds
        Installing Test @ 19:36:19



Flexible IO Tester 3.36:
    pts/fio-2.1.0
    Disk Test Configuration
        1: Random Read
        2: Random Write
```

3: Sequential Read
4: Sequential Write
5: Test All Options
** Multiple items can be selected, delimit by a comma. **
Type: 1,2

1: IO_uring
2: POSIX AIO
3: Sync
4: Linux AIO
5: Test All Options
** Multiple items can be selected, delimit by a comma. **
Engine: 1

1: No
2: Yes
3: Test All Options
** Multiple items can be selected, delimit by a comma. **
Direct: 1

1: 4KB
2: 8KB
3: 16KB
4: 32KB
5: 64KB
6: 128KB
7: 256KB
8: 512KB
9: 1MB
10: 2MB
11: 4MB
12: 8MB
13: Test All Options
** Multiple items can be selected, delimit by a comma. **
Block Size: 1

1: 1
2: 2
3: 4
4: Test All Options
** Multiple items can be selected, delimit by a comma. **
Job Count: 1

System Information

[8192] str_split(): Passing null to parameter #1 ($string) of type string is deprecated in pts_strings:502

PROCESSOR: QEMU Virtual
Core Count: 4

GRAPHICS:

MOTHERBOARD:

MEMORY: 8GB

DISK: 16384GB
File-System: tmpfs

OPERATING SYSTEM: Alpine Linux v3.19 3.19.1
Kernel: 5.10.199-starnix (x86_64)
Compiler: GCC 13.2.1 20231014
System Layer: QEMU

Would you like to save these test results (Y/n):
Enter a name for the result file: starnix-alpine-test1
Enter a unique name to describe this test run / configuration: starnix-alpine-test1

If desired, enter a new description below to better describe this result set / system configuration under test.
Press ENTER to proceed without changes.

Current Description: QEMU testing on Alpine Linux v3.19 3.19.1 via the Phoronix Test Suite.

New Description:

[8192] str_split(): Passing null to parameter #1 ($string) of type string is deprecated in pts_strings:502

[8192] str_split(): Passing null to parameter #1 ($string) of type string is deprecated in pts_strings:502

Flexible IO Tester 3.36:
pts/fio-2.1.0 [Type: Random Read - Engine: IO_uring - Direct: No - Block Size: 4KB - Job Count: 1 - Disk Target: Default Test Directory]

Test 1 of 2
Estimated Trial Run Count: 3
Estimated Test Run-Time: 9 Minutes
Estimated Time To Completion: 17 Minutes [20:05 UTC]
Running Pre-Test Script @ 19:49:24
Started Run 1 @ 19:49:24
The test quit with a non-zero exit status.
Running Interim Test Script @ 19:49:26
Started Run 2 @ 19:49:28
The test quit with a non-zero exit status.
Running Interim Test Script @ 19:49:30
Started Run 3 @ 19:49:32
The test quit with a non-zero exit status.
E: error: failed to setup shm segment
Running Post-Test Script @ 19:49:34
Flexible IO Tester 3.36:
pts/fio-2.1.0 [Type: Random Write - Engine: IO_uring - Direct: No - Block Size: 4KB - Job Count: 1 - Disk Target: Default Test Directory]
Test 2 of 2
Estimated Trial Run Count: 3
Estimated Time To Completion: 9 Minutes [19:57 UTC]
Running Pre-Test Script @ 19:49:42
Started Run 1 @ 19:49:42
The test quit with a non-zero exit status.
Running Interim Test Script @ 19:49:45
Started Run 2 @ 19:49:47
The test quit with a non-zero exit status.
Running Interim Test Script @ 19:49:49
Started Run 3 @ 19:49:51
The test quit with a non-zero exit status.
E: error: failed to setup shm segment
Running Post-Test Script @ 19:49:53

3 Shellbench results

QEMU Alpine container shellbench results:

```
/home/test/shellbench # date; ./shellbench -s sh,bash sample/*; date
Tue Apr 22 18:42:22 EEST 2025
```

name	sh	bash
assign.sh: positional params	1,224,266	382,477
assign.sh: variable	1,559,046	554,859
assign.sh: local var	1,571,926	545,700
assign.sh: local var (typeset)	error	560,615
cmp.sh: [ ]	862,203	305,976
cmp.sh: [[ ]]	error	455,900
cmp.sh: case	1,631,693	542,587
count.sh: posix	1,030,190	414,263
count.sh: typeset -i	error	397,963
count.sh: increment	error	496,585
eval.sh: direct assign	1,142,275	278,549
eval.sh: eval assign	666,838	132,362
eval.sh: command subs	8,537	4,273
func.sh: no func	1,637,581	560,392
func.sh: func	1,336,654	309,703
null.sh: blank	error	error
null.sh: assign variable	1,560,723	587,386
null.sh: define function	1,696,464	574,506
null.sh: undefined variable	1,629,974	446,064
null.sh: : command	1,624,713	552,115
output.sh: echo	902,872	323,873
output.sh: printf	643,064	318,659
output.sh: print	error	error
stringop1.sh: string length	1,187,641	468,215
stringop2.sh: substr 1 builtin	1,285,487	391,393
stringop2.sh: substr 1 echo | cut	2,061	1,444
stringop2.sh: substr 1 cut here doc	3,286	1,692
stringop2.sh: substr 1 cut here str	error	1,598
stringop2.sh: substr 2 builtin	1,041,345	362,014
stringop2.sh: substr 2 echo | cut	1,909	1,406
stringop2.sh: substr 2 cut here doc	3,057	1,607
stringop2.sh: substr 2 cut here str	error	1,584
stringop2.sh: substr 2 expr	2,948	2,344
stringop3.sh: str remove ^ shortest builtin	1,187,400	337,613
stringop3.sh: str remove ^ shortest echo | cut	2,047	1,444
stringop3.sh: str remove ^ shortest cut here doc	3,093	1,511

stringop3.sh:	str remove ^ shortest cut here str	error	1,585
stringop3.sh:	str remove ^ longest builtin	1,193,635	332,950
stringop3.sh:	str remove ^ longest echo | cut	2,072	1,448
stringop3.sh:	str remove ^ longest cut here doc	3,230	1,686
stringop3.sh:	str remove ^ longest cut here str	error	1,680
stringop3.sh:	str remove $ shortest builtin	1,320,233	354,110
stringop3.sh:	str remove $ shortest echo | cut	2,057	1,447
stringop3.sh:	str remove $ shortest cut here doc	3,231	1,709
stringop3.sh:	str remove $ shortest cut here str	error	1,694
stringop3.sh:	str remove $ longest builtin	1,250,991	341,929
stringop3.sh:	str remove $ longest echo | cut	2,062	1,454
stringop3.sh:	str remove $ longest cut here doc	3,152	1,692
stringop3.sh:	str remove $ longest cut here str	error	1,712
stringop4.sh:	str subst one builtin	1,286,484	338,784
stringop4.sh:	str subst one echo | sed	1,992	1,404
stringop4.sh:	str subst one sed here doc	2,959	1,588
stringop4.sh:	str subst one sed here str	error	1,518
stringop4.sh:	str subst all builtin	1,205,907	312,352
stringop4.sh:	str subst all echo | sed	1,878	1,354
stringop4.sh:	str subst all sed here doc	2,940	1,635
stringop4.sh:	str subst all sed here str	error	1,597
stringop4.sh:	str subst front builtin	1,152,285	309,139
stringop4.sh:	str subst front echo | sed	1,922	1,428
stringop4.sh:	str subst front here doc	2,928	1,578
stringop4.sh:	str subst front sed here str	error	1,601
stringop4.sh:	str subst back builtin	1,180,360	312,994
stringop4.sh:	str subst back echo | sed	1,915	1,400
stringop4.sh:	str subst back here doc	2,943	1,608
stringop4.sh:	str subst back sed here str	error	1,600
subshell.sh:	no subshell	1,609,528	509,842
subshell.sh:	brace	1,606,905	514,034
subshell.sh:	subshell	9,181	5,607
subshell.sh:	command subs	8,877	4,918
subshell.sh:	external command	3,813	3,232

* count: number of executions per second
Tue Apr 22 18:50:31 EEST 2025

Starnix Alpine container shellbench results:

/tmp/shellbench-master # date; ./shellbench -s sh,bash sample/*; date
Tue Apr 22 15:53:40 UTC 2025

name	sh	bash
assign.sh: positional params	1,221,626	340,095
assign.sh: variable	1,521,741	529,870

assign.sh: local var	1,533,411	527,202
assign.sh: local var (typeset)	error	514,024
cmp.sh: [ ]	849,187	271,733
cmp.sh: [[ ]]	error	409,341
cmp.sh: case	1,556,708	475,765
count.sh: posix	989,322	404,904
count.sh: typeset -i	error	387,238
count.sh: increment	error	479,394
eval.sh: direct assign	1,144,476	249,909
eval.sh: eval assign	693,713	88,353
eval.sh: command subs	971	732
func.sh: no func	1,637,479	521,087
func.sh: func	1,314,457	292,148
null.sh: blank	error	error
null.sh: assign variable	1,531,410	555,982
null.sh: define function	1,709,268	561,250
null.sh: undefined variable	1,576,157	429,159
null.sh: : command	1,627,484	521,423
output.sh: echo	94,973	76,425
output.sh: printf	62,412	75,892
output.sh: print	error	error
stringop1.sh: string length	1,093,829	443,913
stringop2.sh: substr 1 builtin	1,208,277	371,288
stringop2.sh: substr 1 echo | cut	215	194
stringop2.sh: substr 1 cut here doc	357	221
stringop2.sh: substr 1 cut here str	error	218
stringop2.sh: substr 2 builtin	1,110,112	353,149
stringop2.sh: substr 2 echo | cut	214	191
stringop2.sh: substr 2 cut here doc	360	224
stringop2.sh: substr 2 cut here str	error	222
stringop2.sh: substr 2 expr	389	312
stringop3.sh: str remove ^ shortest builtin	1,142,307	330,134
stringop3.sh: str remove ^ shortest echo | cut	216	192
stringop3.sh: str remove ^ shortest cut here doc	358	221
stringop3.sh: str remove ^ shortest cut here str	error	220
stringop3.sh: str remove ^ longest builtin	1,151,872	321,046
stringop3.sh: str remove ^ longest echo | cut	212	189
stringop3.sh: str remove ^ longest cut here doc	351	215
stringop3.sh: str remove ^ longest cut here str	error	209
stringop3.sh: str remove $ shortest builtin	1,222,173	326,907
stringop3.sh: str remove $ shortest echo | cut	206	185
stringop3.sh: str remove $ shortest cut here doc	361	223
stringop3.sh: str remove $ shortest cut here str	error	220
stringop3.sh: str remove $ longest builtin	1,248,025	335,294
stringop3.sh: str remove $ longest echo | cut	206	190
stringop3.sh: str remove $ longest cut here doc	350	217
stringop3.sh: str remove $ longest cut here str	error	215

stringop4.sh: str subst one builtin	1,194,011	335,010
stringop4.sh: str subst one echo | sed	232	193
stringop4.sh: str subst one sed here doc	407	224
stringop4.sh: str subst one sed here str	error	220
stringop4.sh: str subst all builtin	1,159,633	313,650
stringop4.sh: str subst all echo | sed	226	191
stringop4.sh: str subst all sed here doc	408	222
stringop4.sh: str subst all sed here str	error	213
stringop4.sh: str subst front builtin	1,124,606	306,715
stringop4.sh: str subst front echo | sed	234	187
stringop4.sh: str subst front here doc	408	219
stringop4.sh: str subst front sed here str	error	220
stringop4.sh: str subst back builtin	1,138,948	306,372
stringop4.sh: str subst back echo | sed	236	188
stringop4.sh: str subst back here doc	413	219
stringop4.sh: str subst back sed here str	error	221
subshell.sh: no subshell	1,546,992	499,317
subshell.sh: brace	1,565,604	470,932
subshell.sh: subshell	1,267	831
subshell.sh: command subs	1,010	740
subshell.sh: external command	772	626

* count: number of executions per second
Tue Apr 22 16:01:55 UTC 2025