

Spawn support in pkgsrc tools

Andreas Theodosiou

Google Summer of Code 2016 Project Proposal

The NetBSD Project

19 Mar. 2016

Email: [andreas@atheod.net](mailto:andreas@atheod.net)

## Spawn support in pkgsrc tools

The release of NetBSD 6.0 saw the introduction of the `posix_spawn(3)` system call. `posix_spawn(3)` provides a thread-safe function to create new processes. Currently, pkgsrc tools such as `make` and `sh` use `fork(2)` or `vfork(2)` and an `exec(3)` function to create a child process. During the building of software packages, pkgsrc relies heavily on launching subprocesses. Thus, it would be beneficial to the NetBSD Project to investigate the effect of implementing `posix_spawn(3)` support in the tools that pkgsrc uses to build packages. The aim of this project is to enable pkgsrc to take advantage of the `posix_spawn(3)` system call and compare the efficiency of launching child processes using the `posix_spawn(3)` system call and the current implementation.

### The project

In this section, the scope of the project and its purpose will be analysed and the current state as well as the target, upon completion of the project, state of the pkgsrc tools will be discussed.

As of version 6.0 NetBSD [1] offers support for the `posix_spawn(3)` [2] system call. The package management system used by NetBSD, pkgsrc, builds binary software packages by executing configure scripts and recursively running `make(1)`. This process involves spawning several child processes.

At present, pkgsrc tools spawn child processes using the fork-exec technique. Specifically, the `fork(2)` or `vfork(2)` system call is used to divide the parent process into two identical processes. The newly created child process is an exact copy of the parent process. To overlay the child process with the new, to be run, process, pkgsrc tools use one of the `exec(3)` family functions.

The goal of this project is twofold. Firstly, to implement `posix_spawn(3)` support in `pkgsrc` tools. Secondly, to compare the efficiency of building software packages via `pkgsrc` tools in their current state, using the `fork-exec` technique to spawn child processes, with the efficiency of building packages via an implementation of `pkgsrc` tools that make use of `posix_spawn(3)` to spawn child processes.

As of the date of this project proposal, the NetBSD kernel provides `posix_spawn(3)` support for several computer architectures, also known as ports within the NetBSD project, such as: x86; SPARC64; MIPS; and ARM. The list of ports that provide a `posix_spawn(3)` system call can be found in Table 1.

In order to allow `pkgsrc` to make use of `posix_spawn(3)`, `posix_spawn(3)` support must be implemented primarily in `make(1)` of the base NetBSD system and in shell. While the `make(1)` implementation in NetBSD is fixed to `devel/bmake`, the shell implementation is not as the default installation ships with `sh`, `ksh` and `csh`. Thus it is proposed to start the implementation of the project with one of these shells, specifically `shells/pdksh`. The code changes to `bmake` and `pdksh` should be applied to the upstream versions so as to allow the NetBSD project to adopt the code changes to the base versions of these tools when and as they see fit.

### **List of deliverables**

The required deliverables upon the successful completion of the project are hereby listed:

- `devel/bmake` must be able to launch processes by making use of `posix_spawn(3)`.
- `shells/pdksh` must be able to launch processes by making use of `posix_spawn(3)`.
- Analysis of performance of `pkgsrc` tools with `posix_spawn(3)` implemented in comparison to performance of `fork-exec` versions of said tools.

## Testing procedure

Testing and evaluating the effect of the proposed source code changes will be the most vital aspect of the project. In testing the effect of code changes, one must always keep in mind the user value of implementing said changes.

In the case of `posix_spawn(3)` support in `pkgsrc` tools, the user value is the potential optimisation in the time taken to build packages. Thus in order to correctly evaluate the effectiveness of the project, measurements of the time taken to build different sets of packages and discussion of the results must be performed. The following three test cases are proposed:

1. Build a single package e.g. `editors/ed`.
2. Build a reasonable set of packages.
3. Build the NetBSD 7.0 release.

Data on the three test cases should be gathered both on the current `fork-exec` implementation and for the proposed `posix_spawn()` implementation of `pkgsrc` tools.

In order to evaluate the time to build packages, the `time(1)` tool is to be employed. It is important to highlight the issue of stability when it comes to evaluating data on execution time. Measurements of execution time for the same task are expected to present discrepancy between them due to the fact that different background tasks might be running when each measurement is taken. To counteract this, it is proposed to take several measurements of each test case and calculate the mean value. A minimum of ten measurements for each test case is proposed. In case that the time measurements collected for one of the test cases present a discrepancy that is greater than 10%, the number of measurements will be increased.

After the data has been collected, an evaluation of the effect of implementing `spawn` support in `pkgsrc` tools will be possible.

### **Acceptance criteria**

The following criteria are set as a target to be met upon the completion of the project:

- Spawn implementation of devel/bmake behaves in the same manner as the current implementation of devel/bmake.
- Spawn implementation of shells/pdksh behaves in the same manner as the current implementation of shells/pdksh.
- Performance of pkgsrc tools with spawn support should be comparable to performance without spawn support if not better.

### **Roadmap and high level calendar plan**

Google Summer of Code organizers propose May 23<sup>rd</sup> as the date for students to commence working on their projects. Unfortunately, the Easter Term of my university ends on the second week of June. Consequently, work on this project cannot begin before that time. The roadmap present here takes this constraint into account.

The roadmap is as follows:

- Relevant source code familiarisation and initial experiments. (Mid June)
- Implementation of posix\_spawn(3) support in shells/pdksh (Last week of June)
- Implementation of posix\_spawn(3) support in devel/bmake (Second week of July)
- Test the posix\_spawn(3) implementation of shells/pdksh and devel/bmake to ensure that they are compatible with the fork-exec implementations. (Last week of July).
- Run tests and gather necessary measurements for performance evaluation. (First week of August).
- Carry out analysis of measurements and clean up code (Second week of August).

### **Extension activities**

It is possible to extend the reach of the project to cover implementation of spawn support in other pkgsrc tools, and to re-evaluate the effect of such an implementation on performance.

Possible tools are the following:

- `system(3)`
- `popen(3)`
- NetBSD base sh
- shells/bash

Furthermore, as pkgsrc is available for other operating systems that offer an implementation of the `posix_spawn` system call, an additional extension activity would be to investigate the effect of implementing spawn support in pkgsrc tools for an OS other than NetBSD.

## References

- [1] The NetBSD Project, "Announcing NetBSD 6.0," 17 October 2012. [Online]. Available: <https://www.netbsd.org/releases/formal-6/NetBSD-6.0.html>. [Accessed 18 March 2016].
- [2] The NetBSD Project, "POSIX\_SPAWN(3) - NetBSD Library Functions Manual," 2 February 2014. [Online]. Available: [http://netbsd.gw.com/cgi-bin/man-cgi?posix\\_spawn++NetBSD-7.0](http://netbsd.gw.com/cgi-bin/man-cgi?posix_spawn++NetBSD-7.0). [Accessed 18 March 2016].
- [3] The NetBSD Project, "Platforms Supported by NetBSD," [Online]. Available: <https://netbsd.org/ports/>. [Accessed 18 March 2016].
- [4] The NetBSD Project, "FORK(2) - NetBSD System Calls Manual," 10 June 2004. [Online]. Available: <http://netbsd.gw.com/cgi-bin/man-cgi?fork++NetBSD-7.0>. [Accessed 18 March 2016].
- [5] The NetBSD Project, "EXEC(3) - NetBSD Library Functions Manual," 2014 September 2014. [Online]. Available: <http://netbsd.gw.com/cgi-bin/man-cgi?exec++NetBSD-7.0>. [Accessed 18 March 2016].
- [6] The NetBSD Project, "VFORK(2) - NetBSD System Calls Manual," 18 July 2014. [Online]. Available: <http://netbsd.gw.com/cgi-bin/man-cgi?vfork++NetBSD-7.0>. [Accessed 18 March 2016].
- [7] The NetBSD Project, "MAKE(1) - NetBSD General Commands Manual," 2014 February 2014. [Online]. Available: <http://netbsd.gw.com/cgi-bin/man-cgi?make++NetBSD-7.0>. [Accessed 18 March 2016].

[8] The IEEE and The Open Group, "The Open Group Base Specifications Issue 7 IEEE Std 1003.1," 2013. [Online]. Available:

<http://pubs.opengroup.org/onlinepubs/9699919799/>. [Accessed 18 March 2016].

[9] The NetBSD Project, "Spawn support in pkgsrc tools," [Online]. Available:

[https://wiki.netbsd.org/projects/project/pkgsrc\\_spawn\\_support/](https://wiki.netbsd.org/projects/project/pkgsrc_spawn_support/). [Accessed 18 March 2016].



## Tables

Table 1

*CPU architectures with a `posix_spawn(3)` system call in NetBSD.*

CPU	Tier(s)
alpha	II
arm	I, II, III
hppa	II
m68k	II
powerpc	I, II
sh3	II
sparc	II
sparc64	II
vax	II
x86	I

*Note:* Ports are classified into three tiers based on the current importance of the architecture [3].

The three tiers are the following:

- Tier I: Focus – support is part of NetBSD’s strategy
- Tier II: Organic – evolving at its own pace. In general, these ports boot and work but their maintenance is the responsibility of the user community.
- Tier III: Life Support – severely incapacitated or broken. These ports no longer function.