PROPOSAL

Add support for chdir(2) support in posix_spawn(3)

About the Project

What is the goal of the project?

The main goal of the project is to extend the posix_spawn(3) to include chdir(2) for the newly created process.
posix_spawn() is a POSIX standard method to create a child process which. It takes the necessary arguments, i .e process identifier, name of the file, file actions, spanning attributes, arguments, and environment variables, to spawn a new process.
The requirement here is to change the current working directory using chdir before executing the child process.
As two possible functions for chdir and fchdir will be implemented, it is essential that the prototype of the methods be mentioned in the header files like <spawn.h>, and necessary documentation be provided. This part will happen according to the sequence of actions mentioned in the Austin group bugs tracker.

What will be the deliverables of the project?

Successful implementation of the function to change the working directory will be the key deliverable of this project. The function will also contain the necessary comments and documentation.
Other than that as mentioned in the Austin Groups tracker there will be specific edits in the manual pages and I will also do my best to resolve the issues mentioned relating to posix_spawn_addclose() and posix_spawn_adddup2().

Give an overview of how you intend to reach the project's goal in the form of milestones and a schedule.

This project while not being very hard conceptually, still requires a lot of work. Therefore in my opinion, the first three-four weeks will be about writing new code and integrating it with the already present interfaces. Once this gets done, the next steps will be to test for edge cases to make sure that the functionality implemented is fulfilling the requirements. This part will also take about two to three weeks. To finesse off the project, the code will be cleaned and documented in the final phase of this project.

Is similar software already available elsewhere

Yes in linux, Mac OS and the recent implementation mentioned of the SOLARIS kernel have the ability to change directory when spawning a process using posix_spawn(3).

Is the project a port of software, or a rewrite?

Both posix_spawn and posix_spawnp already exist. But the flexibility of changing the current working directory is absent. Users often exec a shim process, but even that is not in the standard, so users have to come up with that too. The parent process, has the ability to change the director but the child is not provided one. Also because it is expensive to convert relative names into absolute ones, it will be beneficial to extend such a functionality. But as the main posix_spawn functionality already exists, I believe this project could be characterized as a port.

About the Project and NetBSD

If your working area is the core NetBSD operating system: have you installed NetBSD and made first experiences with hands-on configuration? Have you rebuilt the kernel and the userland, either in full or in parts? If you plan to work on pkgsrc, have you installed packages from source and binary? Have you created a package on your own?

The changes required to the files, will happen in the libc functions. Once the required functionality is added, there will be a need to rebuild from source. For that I have already downloaded Virtual Box. In the Virtual Box an instance of the NetBSD OS is already running which was installed through the iso. As I have never built an OS from source ever, I have been reading the relevant articles about it and I believe that in less than a week's time I will be more than comfortable with it.
I have been exploring in the VM for a few days with the installed NetBSD through ssh. As I already have a Mac, getting acquatinted to NetBSD isn't much of a big deal at all.
I also have the source files cloned from the cvs repository, indexed with ctags, which I used for reading the necessary .c files.
Articles being referred to: Chapter 30-34 NetBSD Guide.

Have you found the relevant places that your project is based on in the source code, and read through it?

After going through the series of actions in the Austin Group's defect tracker and the mail I receiver from Martin, I have gone through the, "/src/lib/libc/gen/posix_spawnp.c , posix_spawn_fileactions.c, posix_spawn_shed.c" files. To understand the data structures used in these three files in detail, I have also referred to "/src/lib/libc/include/namespace.h" and "/src/sys/sys/spawn.h". Other than that, while trying to track the implementation of posix_spawn in the kernel I have also gone through "src/external/apache2/llvm/dist/clang/tools/scan-build-py/libear/ear.c" file. I fully understand how the data structures are manipulated and passed to the call_posix_spawn function. In the "src/external/apache2/llvm/dist/clang/tools/scan-build-py/libear/ear.c" file I came across, the manipulation done to the environment variables in the bear_environement and subsequent functions and I fully understand all of it.

How will your project integrate into NetBSD?

This project is a part of the user-land , as the libc functions need to be played with. Once the work is done and checked for edge cases, the changes made, will be submitted in the form of patches.

What interfaces in NetBSD will your project use?

To extend posix_spawn(3) to also include the ability to chdir(2) and fchdir(2), the following probable tasks will be required.

- In file “/usr/src/sys/sys/spawn.h” the enum fae_actions needs two more entries for chdir and fchdir.
- In file “usr/src/lib/libc/gen/posix_spawn_fileactions.c” two more functions corresponding to chdir and fchdir need to be added.
- Once these two are done, we can go ahead and implement the two functions in the “usr/src/lib/libc/gen/posix_spawnp.c” file.

These are the most important tasks that need to be performed.

To what degree are you familiar with those interfaces?

As the data structures that are used in this are pretty basic, like struct, enum and union, I am pretty familiar with them. Also, because I have gone through the list of files mentioned above, I fully comprehend the working and manipulation of the data types and data structures used.

Is knowledge on other topics required for this project, e.g. on hardware, software other than NetBSD, APIs, protocols, etc.? If so, give details and references.

The only knowledge required here is of how a process gets created, and what are the characteristics of address space of the created process. Other than this, a decent background in C programming is necessary. There is no intellectual knowledge of other protocols or hardware direly necessary for this project.

To what degree are you familiar with those? (not/some/very, details?)

I am very familiar with all the requirements.

If the project involves hardware (e.g. writing drivers, doing a port to new hardware, ...): do you own the hardware or have access to?

No hardware is required.

About Me

Can you list some prior projects that you have worked on so far? Include details like programming language, duration, number of people involved, project goal, if you used CVS, SVN or similar, and whatever else we may find thrilling! If you have a CV/resume online, feel free to include a link.

I haven't worked on any big projects per se and I want to change that with this project.
However, I have implemented a few UNIX utilities like, cat, grep, reverse(not standard), and also made a basic shell.
Other than this I have written some basic code implementing a few standard algorithms.
Other than this, I have a very solid background in C and a very thorough knowledge of all the skills required for this project.
This is the link to my resume:
https://www.icloud.com/iclouddrive/0nCDGFcx_A7BfY0--EJO24JuQ#Resume_PiyushSachdeva

Do you have any prior experience with programming NetBSD? In what area? If you did send some problem reports (PRs) or patches, please include references.

I don't have any prior experience working.

Have you previously discussed your project within NetBSD, either on a mailing list or with some specific developers?.

I did send a mail a few days back regarding the project. The mail was sent at "tech-kern@NetBSD.org" and I got a reply from Martin Husemann.
The link from the mailing list is:
http://mail-index.netbsd.org/tech-kern/2021/03/17/msg027157.html

How do we contact you for questions, comments, suggestions, etc.?

My contact details are as follows:
Phone: +91 7895201011 (available on WhatsApp and iMessage)
Email: cosmologist_piyush@icloud.com

Is there anything else you'd like us to know? Did we forget any important details or questions?

The approach I have described in this particular draft is one I thought of on a very superficial level(user-land). However I have also tracked down an implementation of call_posix_spawn in the “src/external/apache2/llvm/dist/clang/tools/scan-build-py/libear/ear.c” file and along with it have read and understood all the functions that eventually get called from this very function. Other than this I have also noticed that the call_posix_spawn() function actually typedefs a function pointer, gets a handle to the symbol “posix_spawn” form the subsequent shared libraries and then it makes the final call. However I have not been able to track any file which is the base implementation(probably in the kernel) of posix_spawn(). I did find the sys call number though.

All I am saying is that if my initial approach is not entirely correct, and something on the above lines needs to be done, then you point me in the direction where I need to look.