Exec

It's also entirely possible for programs to execute other C programs by overlaying themselves with an executable file! This can be done with the `exec` library functions! With the exec functions, you can still pass arguments in just as if you were working through the command line. Let's create two files:

`child.c`

```c
#include<unistd.h>
#include<sys/types.h>
#include<stdio.h>
#include<stdlib.h>

int main(int argc, char** argv) {
  printf("Hello from Child.c!\n");
  printf(" I got %d arguments: \n", argc);

  int i;
  for (i =0; i < argc; i++)
    printf("%s ", argv[i]);

  printf("\nChild is now ending.\n");

  return EXIT_SUCCESS;
}
```

`parent.c`

```c
#include<unistd.h>
#include<sys/types.h>
#include<stdio.h>
#include<stdlib.h>


int main(int argc, char** argv) {
  pid_t childPid = fork();  // This is where the child process splits from the parent

  if (childPid == 0) {
    printf("I am a child! My parent's PID is %d, and my PID is %d\n", getppid(), getpid());

    char* args[] = {"./child", "Hello", "there", "exec", "is", "neat", 0};
```

```
      (args[0], args);

  } else {
    printf("I'm a parent! My pid is %d, and my child's pid is %d \n",
getpid(), childPid);

    sleep(1);

  }

  printf("Parent is now ending.\n");
  return EXIT_SUCCESS;
}
```

So, when we execute this:

```
gcc child.c -o child
gcc parent.c
./a.out
```

> I'm a parent! My pid is 15539, and my child's pid is 15540
>
> I am a child! My parent's PID is 15539, and my PID is 15540
>
> Hello from Child.c!
>
> I got 6 arguments:
>
> ./child Hello there exec is neat
>
> Child is now ending
>
> Parent is now ending.

In the above code, what's happening? So, we're forking off a child process just like we did last time, however, what we're doing immediately after is sending the process to an entirely different executable! Additionally, we're sending it with some arguments to take over! This will be incredibly useful if you ever need to send special flags to your child process's programs! Notice at the end of the array, we have a 0? Try removing that and see what happens! Because we're sending in an array of strings, we need our "string" of arguments to have an end, so we make sure that the new program knows we're done!