Fork, Exec, and I/O redirection
Lab 1

Outline

- Problem statement
- exec() functions
- argc & argv
- fork() & waitpid()
- dup() function

When there is a problem, please raise your hand. Thanks

Any feedback is welcome.

Problem Statement

This lab exercise will help you to understand the concept of fork(), exec() and I/O redirection operations in Linux. This time, we are going to implement a program that executes any command specified by the user and saves the output of the program in a file. We call the program outputsaver. The command line syntax of outputsaver is:

./outputsaver COMMAND OUTPUTFILE

For example, if we type

./outputsaver ls lsoutput.txt

then it will save the output of the command ls into a file called *lsoutput.txt* .

Problem Statement Cont.

Here is one way to implement the outputsaver:

1. fork a child

for the parent:

    3. wait for the child to terminate

for the child:

    3. redirect stdout to the OUTPUTFILE specified in command-line

    4. exec the COMMAND

For this lab, you can assume that the COMMAND takes no additional arguments.

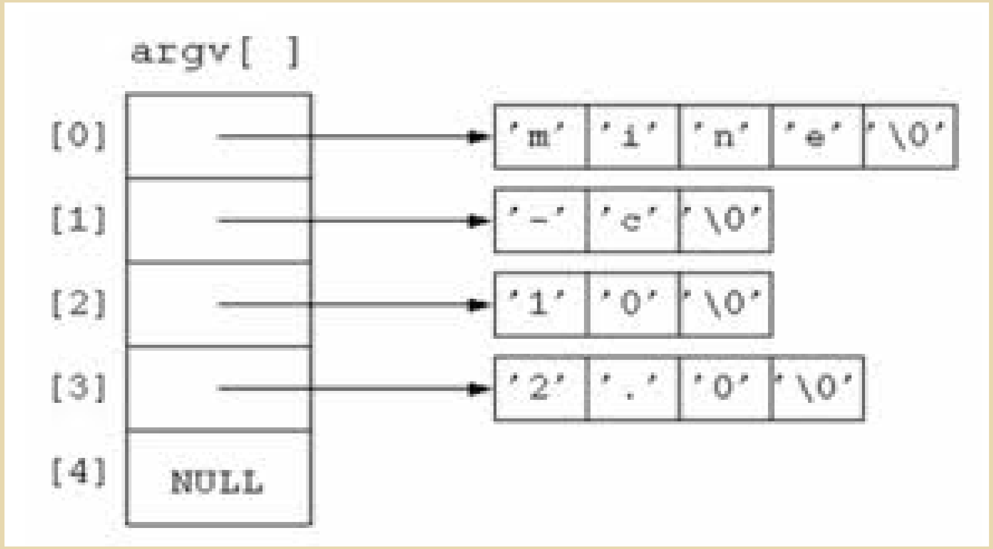
(You get bonus points if you can implement a program that handles arbitrary number of command arguments. For example: ./outputsaver ls -l lsoutput.txt)

exec() functions

- int execl(const char *path, const char *arg, ...);
- int execlp(const char *file, const char *arg, ...);
- int execle(const char *path, const char *arg, ..., char * const envp[]);
- int execv(const char *path, char *const argv[]);
- int execvp(const char *file, char *const argv[]);

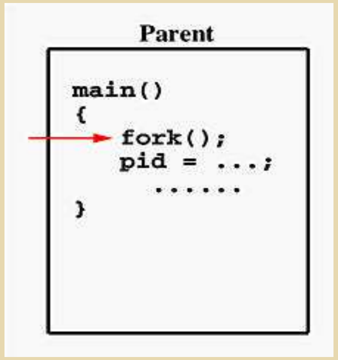
argv argc

mine -c 10 2.0



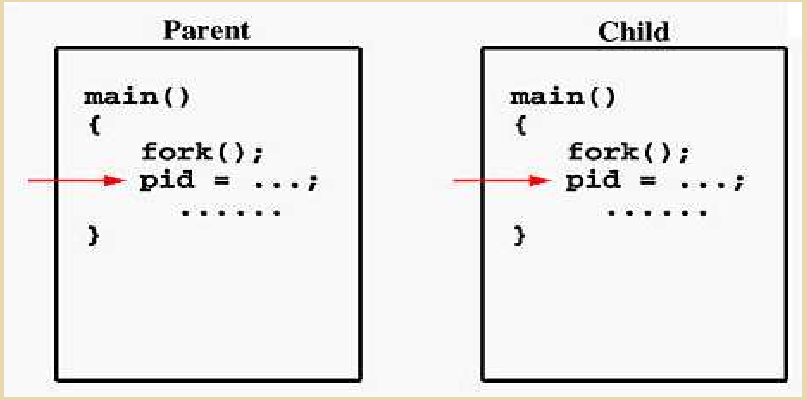
fork() & waitpid()

Step 1



```
main()
{
   fork();
   pid = ...;
      ......
}
```

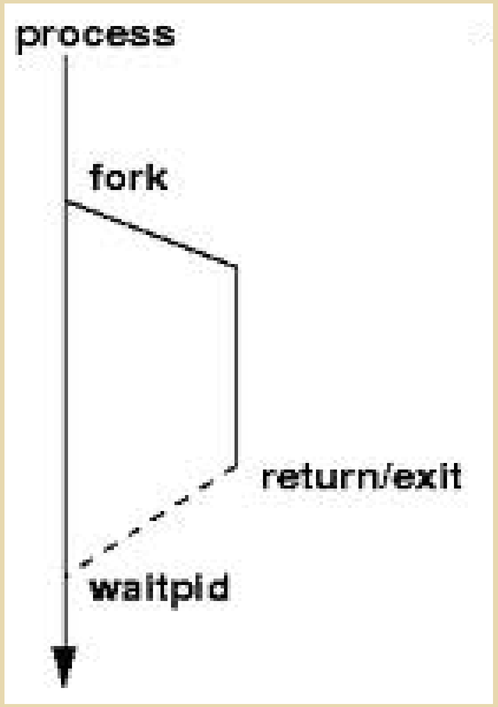
Step 2



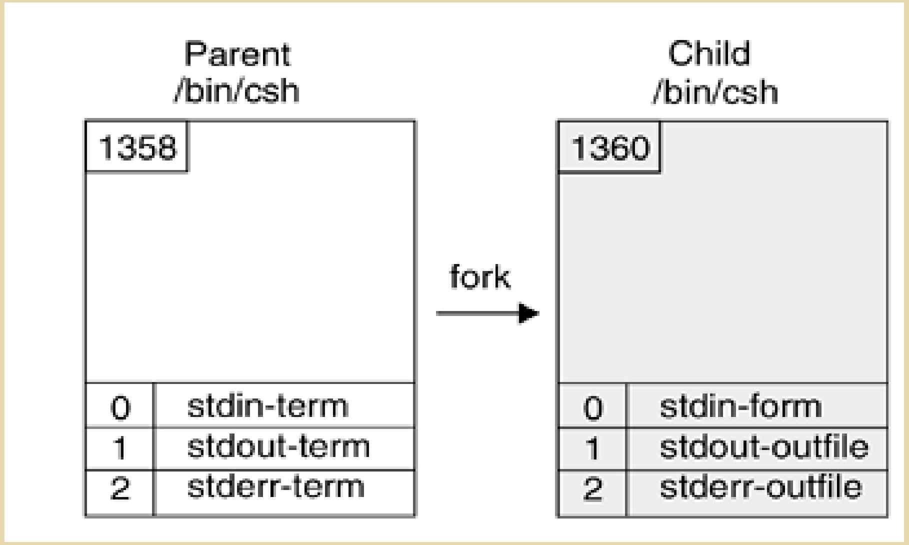
```
main()
{
   fork();
   pid = ...;
      ......
}
```

```
main()
{
   fork();
   pid = ...;
      ......
}
```

fork() & waitpid() cont.



dup2()



Steps to change stdout

•After fork(), in child process, open the outfile

•Call dup(), duplicate the outfile descriptor to stdout

•Execute child binary

dup2() example

```
/* dup2ex.c
   Jim Plank
   CS360
   Dup lecture */

#include <stdio.h>
#include <fcntl.h>

main()
{
  int fd;
  char *s;

  fd = open("file4", O_WRONLY | O_CREAT | O_TRUNC, 0666);

  if (dup2(fd, 1) < 0) { perror("dup2"); exit(1); }

  printf("Standard output now goes to file4\n");

  close(fd);

  printf("It goes even after we closed file descriptor %d\n", fd);

  putchar('p');
  putchar('u');
  putchar('t');
  putchar('c');
  putchar('h');
  putchar('a');
  putchar('r');
  putchar(' ');
  putchar('w');
  putchar('o');
  putchar('r');
  putchar('k');
  putchar('s');
  putchar('\n');

  s = "And fwrite\n";

  fwrite(s, sizeof(char), strlen(s), stdout);

  fflush(stdout);

  s = "And write\n";
  write(1, s, strlen(s));
}
```

Thanks