

Structure of a command shell

--writing our own, homemade but full-fledged

David Morgan

Capabilities of the “real” shell

- Command processing
 - parse
 - expand
 - execute
- I/O redirection
- Piping
- Environment control
- Background processing
- Shell scripts

© David Morgan 2005-18

The dispensible ones (☒)

- Command processing
 - pa☒e
 - ex☒nd
 - **execute** ← the single essential, for a “command” shell
- I/O☒redirection
- Pi☒ng
- En☒ironment control
- Ba☒kground processing
- Sh☒l scripts

© David Morgan 2005-18

Review: process spawns process the fork/exec formula (fork5 spawns ls)

```
root@EMACH1:~/class/books/molay/ch08/bookcode
File Edit View Terminal Tabs Help
[root@EMACH1 bookcode]# cat fork5.c
#include <unistd.h>
#include <stdio.h>
main() { int result;
  printf( "\nParent does stuff and then...\n\n" );
  result = fork();
  if ( result == 0 ) {
    printf("Child could run some executable...\n\n");
    execl("/bin/ls", "/bin/ls", "-l", "/etc/httpd/conf/", NULL); }
  else
    printf("...parent do something completely different.\n\n"); }

[root@EMACH1 bookcode]# ./fork5
Parent does stuff and then...

Child could run some executable...

total 60
-rw-r--r--  1 root root 32809 May 23 05:14 httpd.conf } ls -l /etc/httpd/conf
-rw-r--r--  1 root root 12958 May 23 05:14 magic      (the real thing)
...parent do something completely different.

[root@EMACH1 bookcode]#
```

© David Morgan 2005-18

To make a “shell” out of it...

- make it orderly timing discipline
- make it interactive user involvement
- make it repetitive as long as he wants

© David Morgan 2005-18

Some system function calls

- **fork** - creates a child process that differs from the parent process only in its PID and PPID
- **exec** - replaces the current process image with a new process image
- **wait** - suspends execution of the current process until its child has exited
- **exit** - causes normal program termination and a return value sent to the parent

© David Morgan 2005-18

Make it orderly

insert wait (parent) & exit (child)

```
root@EMACH1:~/class/books/molay/ch08/bookcode - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@EMACH1 bookcode]# cat fork6.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
main() { int result;
  printf( "\nParent does stuff and then...\n\n" );
  result = fork();
  if ( result == 0 )
  {
    printf("Child could run some time-consuming executable...\n\n");
    execl("timewaster", "timewaster", NULL);
    exit(0);
  }
  else
  {
    printf("...while parent might plunge ahead in parallel...\n");
    wait(NULL);
    printf("...and/or wait for child termination to continue.\n\n");
  }
}

[root@EMACH1 bookcode]# ./fork6

Parent does stuff and then...

Child could run some time-consuming executable...

...while parent might plunge ahead in parallel...
...and/or wait for child termination to continue.
```

Annotations in the image:

- Red arrow pointing to `wait(NULL);`: **tell parent when done**
- Red arrow pointing to `execl("timewaster", "timewaster", NULL);`: **prog that wastes 10 sec**
- Red arrow pointing to the output line `...and/or wait for child termination to continue.`: **here, 10 sec passes**
- Red text above the code: **stall till child is done**

05-18

Make it interactive

insert scanf() for program name

```
root@EMACH1:~/class/books/molay/ch08/bookcode - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@EMACH1 bookcode]# cat fork8.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
main() { int result;
  char prog[100];
  printf( "\nParent does stuff and then...\n\n" );
  result = fork();
  if ( result == 0 )
  {
    printf("Child could run a user-specified executable:\n");
    printf("What executable should the child run? ");
    scanf( "%s", &prog);
    execl(prog,prog,NULL);
    exit(0);
  }
  else
  {
    wait(NULL);
    printf("\n...parent continues now, only when child is done.\n\n");
  }
}; }

[root@EMACH1 bookcode]# ./fork8

Parent does stuff and then...

Child could run a user-specified executable:
What executable should the child run? /bin/date
Tue Oct 25 22:32:25 PDT 2005

...parent continues now, only when child is done.
```

Annotations in the image:

- Red circle around the child's code block: `printf("Child could run a user-specified executable:\n"); printf("What executable should the child run? "); scanf("%s", &prog); execl(prog,prog,NULL); exit(0);`
- Red arrow pointing to the user input `/bin/date` in the output.

2005-18

Make it repetitive

stick it in a while () loop

```
root@EMACH1:~/class/books/molay/ch08/bookcode - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@EMACH1 bookcode]# cat fork9.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
main()
{
    int result;
    char prog[100];

    printf( "\nWelcome to the Dshell...\n\n" );
    while (1)
    {
        printf("What executable should the child run? ");
        scanf( "%s", &prog);
        result = fork();
        if ( result == 0 )
        {
            execl(prog,prog,NULL);
            exit(0);
        }
        else
        {
            wait(NULL);
            printf("\n...done waiting. Welcome back!\n-----\n\n");
        }
    }
}

[root@EMACH1 bookcode]# ./fork9

Welcome to the Dshell...

What executable should the child run? /bin/date
Tue Oct 25 22:40:13 PDT 2005

...done waiting. Welcome back!
-----

What executable should the child run? /bin/pwd
/root/class/books/molay/ch08/bookcode

...done waiting. Welcome back!
-----

What executable should the child run? █
```

© Morgan 2005-18

Now install it as a default shell

```
root@EMACH1:~/class/books/molay/ch08/bookcode - Shell - Konsole
Session Edit View Bookmarks Settings Help

[root@EMACH1 bookcode]# cp fork9 /bin/dsh ← call it "dsh"
[root@EMACH1 bookcode]# chmod 755 /bin/dsh
[root@EMACH1 bookcode]# useradd -s /bin/dsh rudi ← give it to "rudi"
[root@EMACH1 bookcode]#
[root@EMACH1 bookcode]# cat /etc/passwd | grep rudi
rudi:x:531:531:./home/rudi:/bin/dsh
[root@EMACH1 bookcode]#
[root@EMACH1 bookcode]# su rudi ← become "rudi" to invoke his shell

Welcome to the Dshell...

What executable should the child run? /usr/bin/whoami
rudi

...done waiting. Welcome back!
-----

What executable should the child run? █
```

© David Morgan 2005-18

Limitations of this 15-line shell

- full-path commands only
- single-token commands only (no args!)
- no frilly bash stuff
 - no filename globbing (wildcarding)
 - no history (commandline recall)
 - no redirection/pipes
 - no variables
 - no builtins (cd, exit, etc)

© David Morgan 2005-18

You gotta problem with it??

- terms are open source-- you can outcode me
- have it run commands that *do* give frilly stuff
 - /bin/ksh
 - /bin/tcsh
 - /bin/zsh
 - /bin/bash

© David Morgan 2005-18