Process and process Management part 4

Agenda

system();

perror();

fork();

System() function

- The system() function allows the calling program to execute a shell command.

```
int system( const char *command );
```

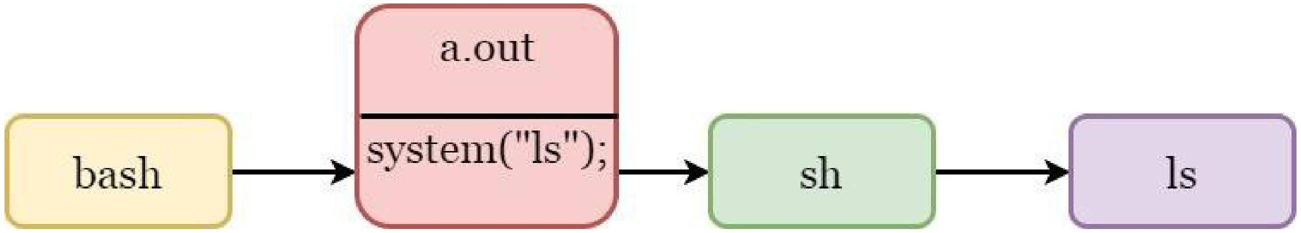
- The system() function creates a child process that invokes a shell to execute command. Here is an example of a call to system():

```
system(“ls -l”);
```

- system() executes a command specified in the brackets by calling `/bin/sh -c command`, *returns* after the command has been completed.
- It returns -1 on error

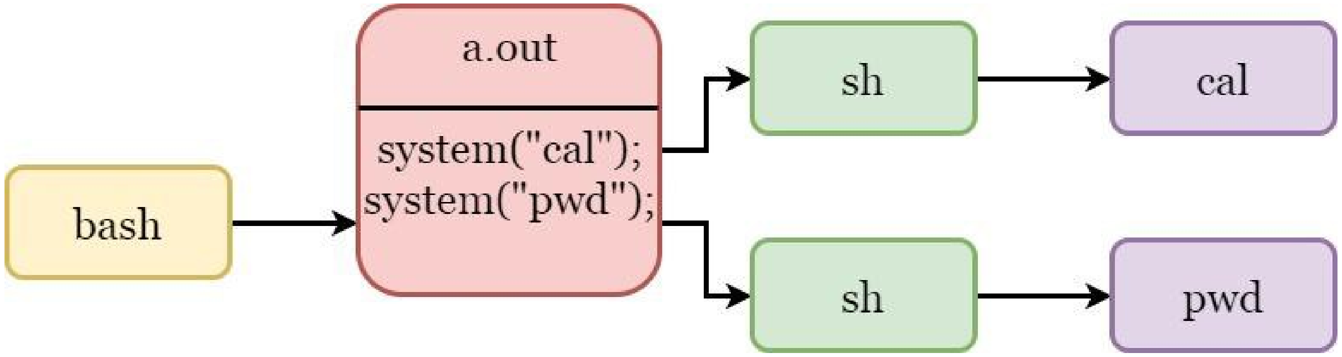
System() function (Contd..)

```
#include <stdio.h>
 main()
 {
printf(“Hello\n”);
system(“ls”);
printf(“Hi”);
 }
```



System() function (Contd..)

```
#include <stdio.h>
main()
{
printf(“Hello\n”);
system(“cal”);
system(“pwd”);
printf(“Hi”);
 }
```



Perror() function

```
void perror ( const  char  *s );
```

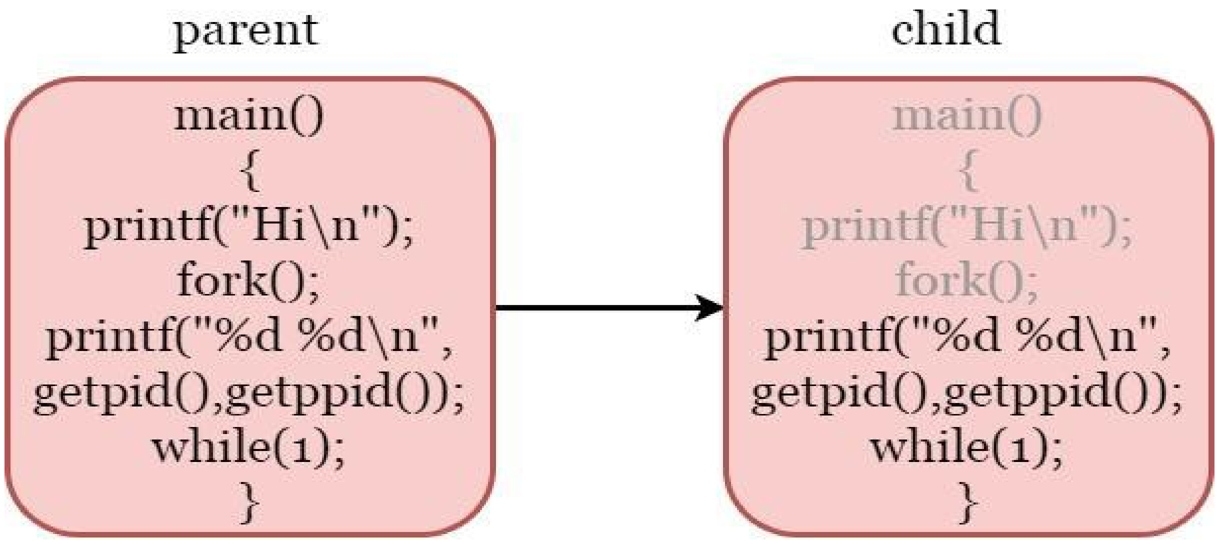
- The perror() function produces a message on standard error describing the last error encountered during a call to a system or library function.
- First (if s is not NULL and *s is not a null byte ('\0')), the argument string s is printed, followed by a colon and a blank. Then an error message corresponding to the current value of errno and a new-line.
- The <errno.h> header file defines the integer variable errno, which is set by system calls and some library functions in the event of an error to indicate what went wrong.

Fork() Funtion

- fork() creates a new process by duplicating the calling process.
  The new process is referred to as the child process.
  The calling process (who is calling fork()) is referred to as the parent process.
- The child process is an exact duplicate of the parent process.
- The child process and the parent process run in separate memory spaces.

Fork() Function (Contd..)

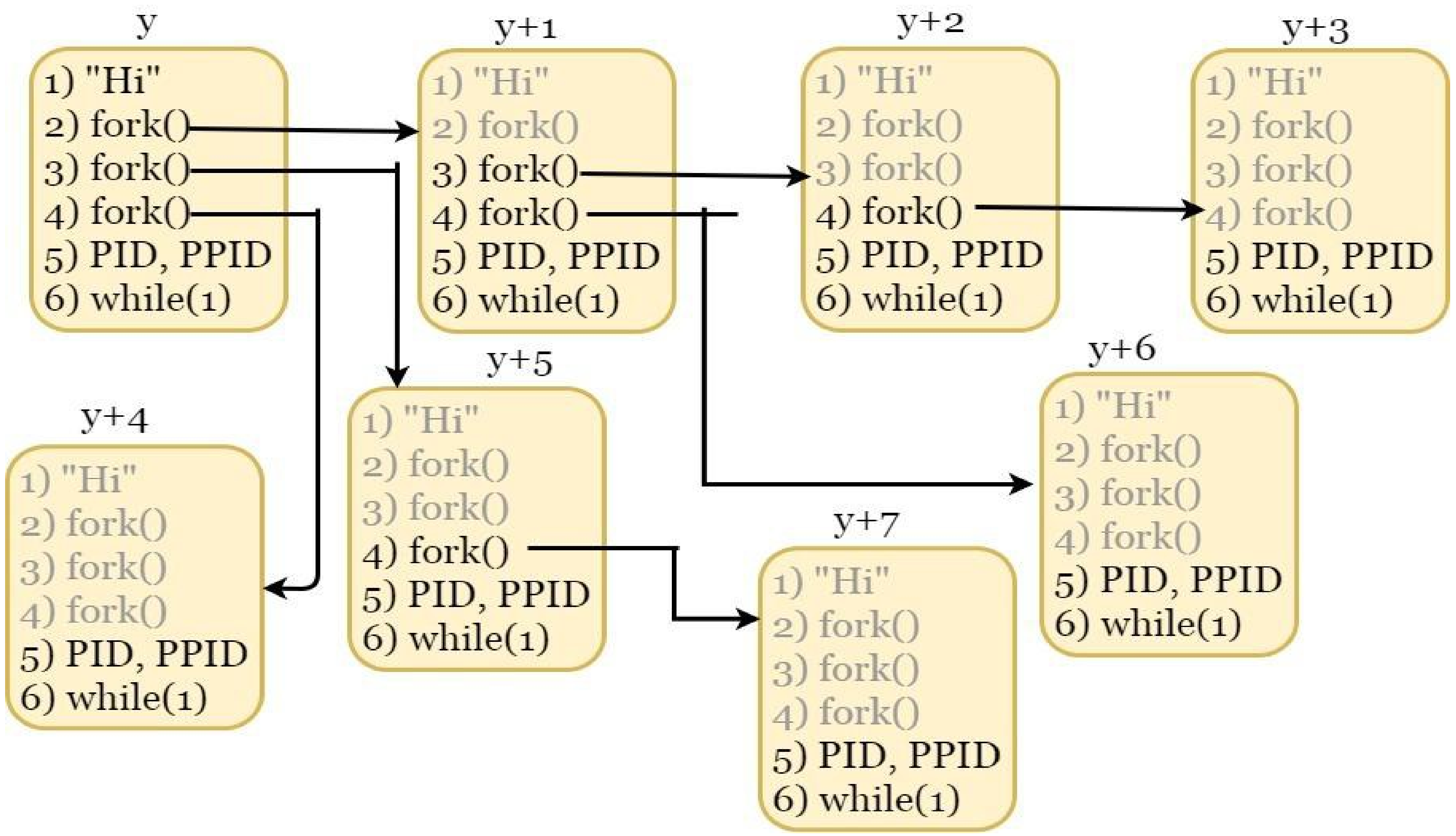
```
main() {
printf("Hi...\n");
 fork();    //creates child process
printf("PID:%d, PPID:%d\n",getpid(), getppid());
/* this is executed twice */
while(1);  }
```



Fork() Function (Contd..)

```
main()
{
 printf("Hi...\n");
 fork();
 fork();
  fork();
 printf("PID:%d, PPID:%d\n",getpid(), getppid());
while(1);
 }
```

Fork() Function (Contd..)



Fork() Function (Contd..)

- On success, the PID of the child process is returned in the parent, and 0 is returned in the child.

- On failure, -1 is returned in the parent, no child process is created,and errno is set appropriately.