LAB - Process Operations

Dr. Biswajeet Sethi

fork()

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{

        // make two process which run same
        // program after this instruction
        pid_t p = fork();
        if(p<0){
        perror("fork fail");
        exit(1);
        }
        printf("Hello world!, process_id(pid) = %d \n",getpid());
        return 0;
}
```

exec()

```
// EX1.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
	printf("PID of ex1.c = %d\n", getpid());
	char *args[] = {"Hello", "Neso", "Academy", NULL};
	execv("./ex2", args);
	printf("Back to ex1.c");
	return 0;
}
```

```
//EX2.c

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
	printf("We are in ex2.c\n");
	printf("PID of ex2.c = %d\n", getpid());
	return 0;
}
```

Wait ()

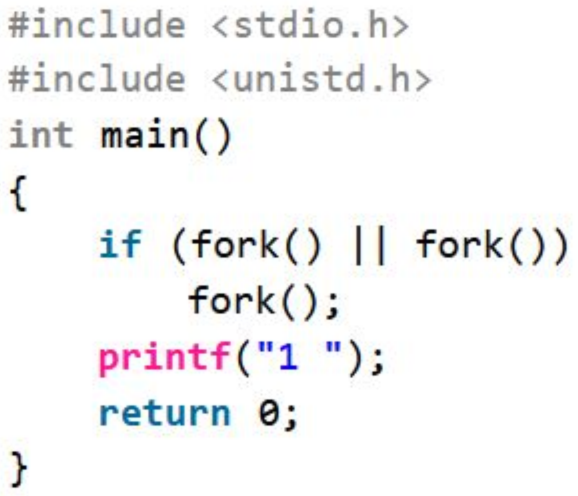
```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
        pid_t pid;

        pid = fork();

        if (pid == 0) { // Child process
        printf("Child process with PID: %d\n", getpid());
        exit(0); // Terminate child
        } else if (pid > 0) { // Parent process
        wait(NULL); // Wait for child to finish
        printf("Parent process with PID: %d, child exited with PID: %d\n", getpid(), pid);
        } else {
        perror("fork failed");
        exit(1);
        }

        return 0;
}
```

SOME MORE EXAMPLES

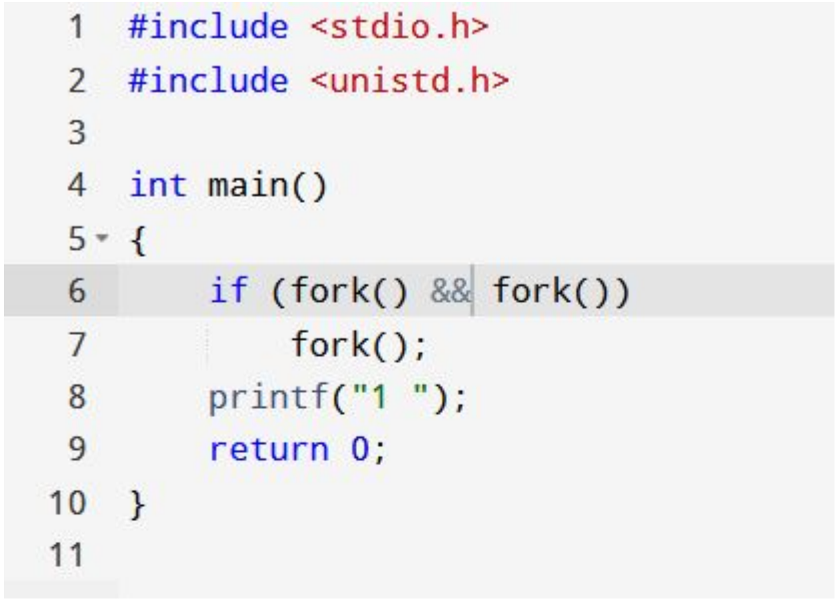
IN

fork()

Find the o/p !



```
#include <stdio.h>
#include <unistd.h>
int main()
{
    if (fork() || fork())
        fork();
    printf("1 ");
    return 0;
}
```



```
#include <stdio.h>
#include <unistd.h>

int main()
{
    if (fork() && fork())
        fork();
    printf("1 ");
    return 0;
}

```

Find the o/p !!

```
#include <stdio.h>
void main()
{
   int i;
   for (i=0;i<3;i++)
   {
      fork();
      // getppid(): gets the parent process-id
      // getpid(): get child process-id

      printf("[%d] [%d] i=%d\n", getppid(), getpid(), i);
   }

   printf("[%d] [%d] hi\n", getppid(), getpid());
}
```

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
        fork();
        fork();
        fork();
        printf("hello\n");
        return 0;
}
```

```
#include <stdio.h>
#include <unistd.h>
int main()
{
        fork();
        fork() && fork() || fork();
        fork();

        printf("forked\n");
        return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
void forkexample()
{
	pid_t p;
	p = fork();
	if(p<0)
	{
	perror("fork fail");
	exit(1);
	}
	// child process because return value zero
	else if ( p == 0)
	printf("Hello from Child!\n");

	// parent process because return value non-zero.
	else
	printf("Hello from Parent!\n");
}
int main()
{
	forkexample();
	return 0;
}
```