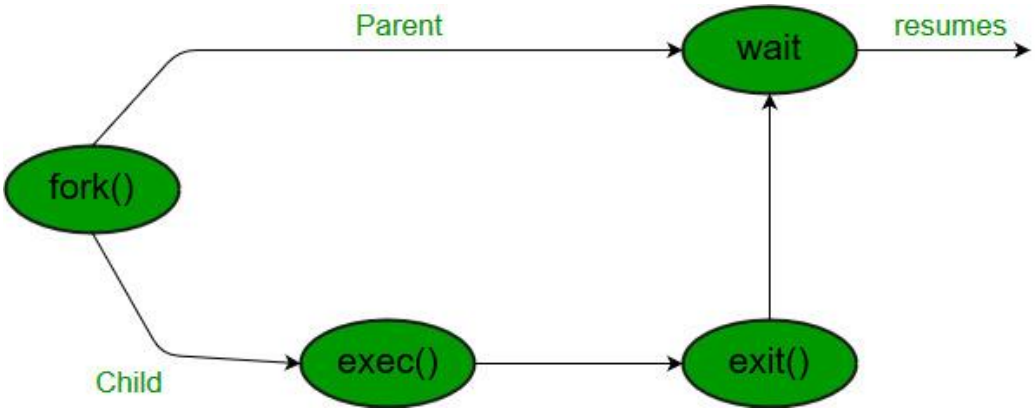
Introducción a System Calls con wait()

Una llamada a wait() bloquea el proceso hasta que uno de sus procesos hijos finalice o se reciba una señal. Después de que el proceso hijo finaliza, el padre *continúa* su ejecución después de la instrucción wait().

El proceso hijo puede terminar debido a cualquiera de estas situaciones:

- Llama a exit();
- Devuelve un int desde main()
- Recibe una señal (del sistema operativo u otro proceso) cuya acción predeterminada es terminar.



Consideraciones:

- Si un proceso no tiene hijos, entonces wait() devuelve inmediatamente -1

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

int main()
{
  int pid = getpid();
  printf( "Iniciando proceso con pid %d\n", pid );
  printf( "Llamando a wait\n" );
  int estatus = wait( pid );
  printf( "Despues de wait\n" );
  printf( "Estatus de retorno de wait=%d\n",estatus );
}
```

wait1.c

- Si solo se finaliza un proceso hijo, entonces wait () devuelve el ID del proceso hijo terminado.

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

int main()
{
  int retorno_fork;
  int arg_wait;
  int pid = getpid();
  printf( "Iniciando proceso con pid %d\n", pid );
  printf( "Llamando a fork\n" );
  retorno_fork = fork();
  if( retorno_fork != 0 )
  {
    printf( "Proceso padre, la variable
retorno_fork=%d\n",retorno_fork );
    printf( "Proceso padre, llamando a wait\n" );
    int retorno_wait =  wait( &arg_wait );
    printf( "Proceso padre, despues de wait\n" );
    printf( "Proceso padre, el valor &arg_wait=%d\n",&arg_wait );
    printf( "Proceso padre, la variable arg_wait=%d\n",arg_wait );
    printf( "Proceso padre, la variable
retorno_wait=%d\n",retorno_wait );
  }
  else
  {
    printf( "Proceso hijo, el pid es %d\n", getpid() );
    printf( "Proceso hijo, a dormir 10 seg\n" );
    sleep( 10 );
    printf( "Proceso hijo, despierta\n" );
  }
}
```

wait2.c

- Si se finalizan más de un proceso hijo, wait() recupera cualquier *hijo arbitrariamente* y devuelve un ID de proceso de ese proceso hijo.

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

int main()
{
  int retorno_fork;
  int arg_wait;
  int pid = getpid();
  printf( "Iniciando proceso con pid %d\n", pid );
  printf( "Llamando a fork la primera vez\n" );
  retorno_fork = fork();
  if( retorno_fork != 0 )
  {
    printf( "Proceso padre, la variable retorno_fork la primera
vez=%d\n",retorno_fork );
    printf( "Llamando a fork la segunda vez\n" );
    retorno_fork = fork();
    if( retorno_fork != 0 )
    {
      printf( "Proceso padre, llamando a wait\n" );
      int retorno_wait =  wait( &arg_wait );
      printf( "Proceso padre, despues de wait\n" );
      printf( "Proceso padre, el valor &arg_wait=%d\n",&arg_wait
);
      printf( "Proceso padre, la variable arg_wait=%d\n",arg_wait
);
      printf( "Proceso padre, la variable
retorno_wait=%d\n",retorno_wait );
    }
    else
    {
      printf( "Proceso hijo 2, el pid es %d\n", getpid() );
      printf( "Proceso hijo 2, a dormir 10 seg\n" );
      sleep( 10 );
      printf( "Proceso hijo 2, despierta\n" );
    }
  }
  else
  {
    printf( "Proceso hijo 1, el pid es %d\n", getpid() );
    printf( "Proceso hijo 1, a dormir 10 seg\n" );
    sleep( 10 );
    printf( "Proceso hijo 1, despierta\n" );
  }
}
```

wait3.c