Auxiliar 10 - Fork y pipes

Profesor: Luis Mateu
Auxiliares: Gerard Cathalifaud
Vicente González
Joaquín López
Rodrigo Urrea

1 Resumen

La creación de procesos pesados en Unix es usando la llamada *fork*

- int fork(): crea un proceso clon del que lo invocó, quien lo llama toma el rol de padre y retorna el pid correspondiente al hijo (número entre 1 y 100) mientras que el hijo verá como resultado retornado el 0 (es un indicador, no corresponde realmente a su identificador). En la librería "unistd.h"
- El proceso hijo recibe una copia exacta de la memoria del padre, pero son espacios independientes, por lo que modificaciones realizadas por el hijo no afecta a la memoria del padre y en viceversa. Tener en cuenta que el hijo hereda los files descriptors abiertos.
- void exit(int status): si se ejecuta en el proceso hijo, este muere con el código de retorno
- pid_t waitpid(pid_t pid, int *status, int options): Ejecutar con opción 0, para esperar que el proceso (hijo) termine. En la librería "sys/wait.h"
- int WEXITSTATUS(int *status) entrega los últimos 8 bits del código de retorno.
- mientras que el hijo termina con exit, el padre debe enterrarlo con waitpid() (para así evitar que se quede ocupando memoria y procesador (zombie)) Si el pid que se entrega a waitpid es -1, espera a que termine cualquiera de los pid hijos y entrega su pid.
- void pipe(int *fds): Un arreglo de tamaño 2, donde fds[0] corresponde a la lectura (read) del archivo y fds[1] a la escritura (write). (pipe es equivalente a | en la consola).

P1. Búsqueda del nodo

Cree un programa que, usando *fork*, encuentre el nodo que contiene un valor en un árbol binario no ordenado. Se debe realizar una búsqueda exhaustiva, piense en la estrategia, divide y conquista. A continuación se muestra un código base:

```
typedef struct nodo {
  char *val;
  struct nodo *izq, *der;
} Nodo;
Nodo *buscar(Nodo *a, char *val, char *p)
```

P2. Control 3 2018-2 P1 a)

Se ha programado secuencialmente la función *suma* de la siguiente manera:

```
int suma(double x0, double dx, int n, double *pres) {
 double s= 0;
 for (int k= 0; k<n; k++)
    s+= f(g(x0+dx*k));
 *pres= s;
 return 0;
}
```

Cada evaluación de f y g es lenta y por eso se requiere paralelizar. Reescriba esta función paralelizándola para una máquina dual-core. Para ello use una sola vez la llamada al sistema *fork*. El proceso hijo debe realizar todas las evaluaciones de la función g, enviando sus resultados al padre por medio de un *pipe*. El padre realiza todas las evaluaciones de la función f tomando como argumento los resultados calculados por el hijo. El resultado final retornado por la función *suma* debe ser el mismo calculado por su versión original. No olvide enterrar adecuadamente al hijo.