Usando *fork()*

Sistemas Operativos

Departamento de Computación, FCEyN,
Universidad de Buenos Aires, Buenos Aires, Argentina

3 de Septiembre de 2019

Clase de hoy

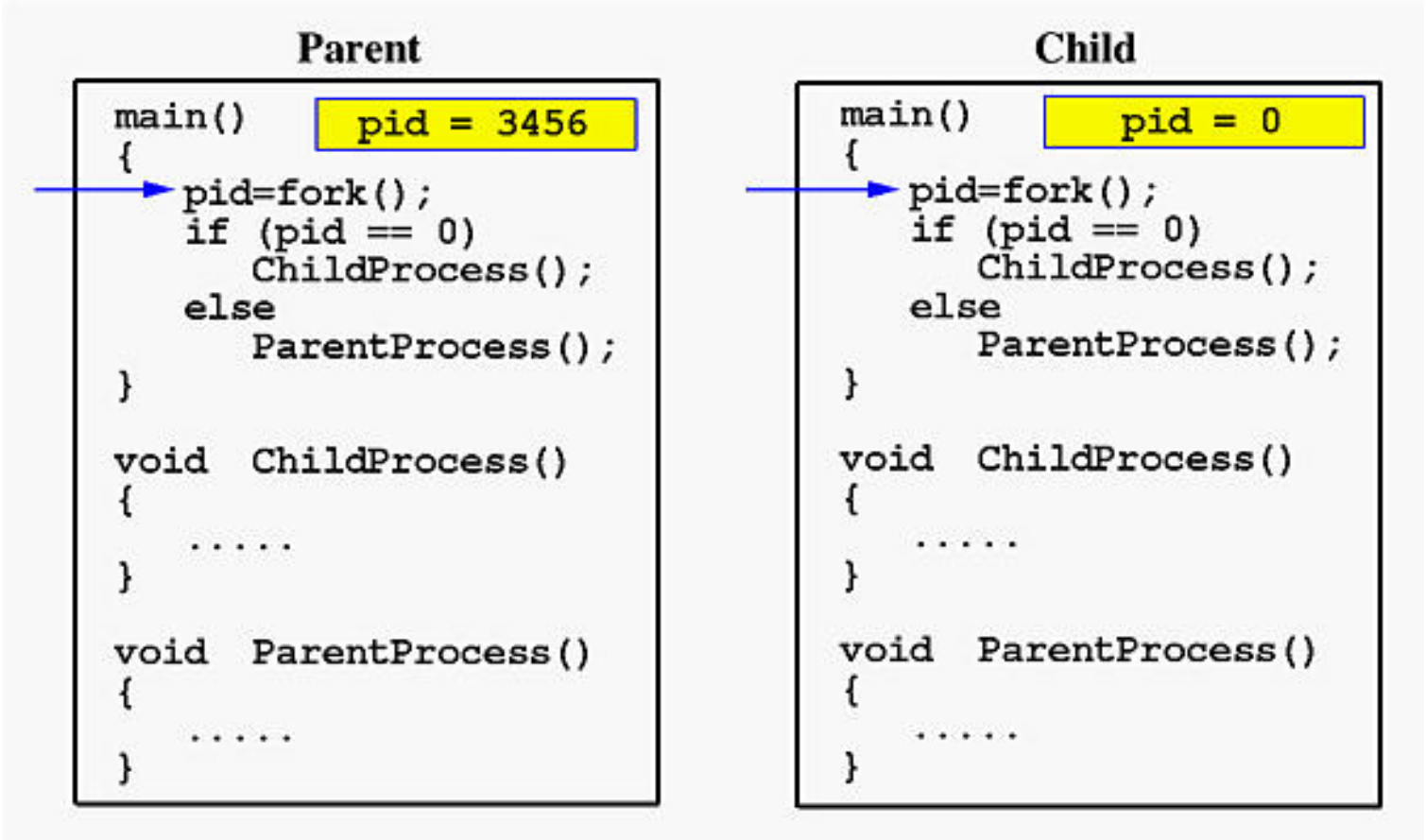
- Cómo funciona Fork?
- Cómo maneja los recursos?
- Qué problemas tiene?
- Ejercicio “Por Discord”

Creación y control de procesos

- `pid_t fork(void):` Crea un nuevo proceso.
  En el caso del creador (padre) se retorna el process id del hijo. En caso del hijo, retorna 0.

Creación de procesos utilizando *fork*

El siguiente programa crea un proceso nuevo.
Luego, implementa funciones distintas para el creador del proceso y el hijo.



Creación y control de procesos

- `pid_t fork(void)`: Crea un nuevo proceso.
  En el caso del creador (padre) se retorna el process id del hijo. En caso del hijo, retorna 0.

Creación y control de procesos

- `pid_t fork(void)`: Crea un nuevo proceso.
  En el caso del creador (padre) se retorna el process id del hijo. En caso del hijo, retorna 0.
- `int execve(const char *filename, char *const argv[], char *const envp[])`: Sustituye la imagen de memoria del programa por la del programa ubicado en filename.

Creación y control de procesos

- `pid_t fork(void)`: Crea un nuevo proceso.
  En el caso del creador (padre) se retorna el process id del hijo. En caso del hijo, retorna 0.
- `int execve(const char *filename, char *const argv[], char *const envp[])`: Sustituye la imagen de memoria del programa por la del programa ubicado en filename.
- `pid_t vfork(void)`: Crea un hijo sin copiar la memoria del padre, el hijo tiene que hacer exec.

Creación y control de procesos

- `pid_t wait(int *status)`: Bloquea al padre hasta que el hijo termine (si no se indica ningún status) o hasta que el hijo alcance el estado indicado.

Creación y control de procesos

- `pid_t wait(int *status)`: Bloquea al padre hasta que el hijo termine (si no se indica ningún status) o hasta que el hijo alcance el estado indicado.
- `pid_t waitpid(pid_t pid, int *status, int options)`: Igual a wait pero espera al proceso correspondiente al pid indicado.

Creación y control de procesos

- `pid_t wait(int *status)`: Bloquea al padre hasta que el hijo termine (si no se indica ningún status) o hasta que el hijo alcance el estado indicado.
- `pid_t waitpid(pid_t pid, int *status, int options)`: Igual a wait pero espera al proceso correspondiente al pid indicado.
- `void exit(int status)`: Finaliza el proceso actual.

Creación y control de procesos

- `pid_t wait(int *status)`: Bloquea al padre hasta que el hijo termine (si no se indica ningún status) o hasta que el hijo alcance el estado indicado.
- `pid_t waitpid(pid_t pid, int *status, int options)`: Igual a wait pero espera al proceso correspondiente al pid indicado.
- `void exit(int status)`: Finaliza el proceso actual.
- `int clone(...)`: Crea un nuevo proceso. El hijo comparte parte del contexto con el padre. Es usado en la implementación de threads.

Ejemplo: Creación de procesos (fork)

```
1  int main(void) {
2    pid_t pid = fork();
3    int a=1;
4    if (pid == -1) exit(EXIT_FAILURE);
5    //si es -1, hubo un error
6    else if (pid == 0) {
7      for(int i=0; i<11;i++){
8          a++;
9      }
10   }  else {
11      waitpid(pid);
12      print(i);
13   }
14 }
```

Ejemplo: Creación de procesos (fork)

```
1  int main(void) {
2    pid_t pid = fork();
3    int a=1;
4    if (pid == -1) exit(EXIT_FAILURE);
5    //si es -1, hubo un error
6    else if (pid == 0) {
7      for(int i=0; i<11;i++){
8          a++;
9      }
10   }  else {
11      waitpid(pid);
12      print(i);
13   }
14 }
```

Que imprime?

Que pasa con la memoria?

- Los procesos no comparten memoria cada uno cuenta con su propio espacio de memoria para usar.

Que pasa con la memoria?

- `Los procesos no comparten memoria cada uno cuenta con su propio espacio de memoria para usar.`

  ¿Y si el proceso esta usando 3Gb de memoria? ¿No es muy pesado copiar toda la memoria de un proceso a otro?

Que pasa con la memoria?

- `Los procesos no comparten memoria cada uno cuenta con su propio espacio de memoria para usar.`

  ¿Y si el proceso esta usando 3Gb de memoria? ¿No es muy pesado copiar toda la memoria de un proceso a otro?
- `Un proceso es creado usando fork() comienza con sus paginas de memoria apuntan a las mismas que el padre.`

Que pasa con la memoria?

- Los procesos no comparten memoria cada uno cuenta con su propio espacio de memoria para usar.

  ¿Y si el proceso esta usando 3Gb de memoria? ¿No es muy pesado copiar toda la memoria de un proceso a otro?
- Un proceso es creado usando fork() comienza con sus paginas de memoria apuntan a las mismas que el padre.
- Recien cuando el padre o el hijo escriban en memoria, se hace finalmente la copia.
- Esto se le llama **Copy-on-Write** .

Usando fork, que puede pasar?

¿Cuantos procesos se van a correr en total?

```
1  int main(void) {
2    fork();
3    fork();
4    fork();
5  }
6
```

Usando fork, que puede pasar?

Ahora, que creen que va a pasar en este caso?

```
1  int main(void) {
2    while(1){
3      fork();
4    }
5  }
```

Usando fork, que puede pasar?

Ahora, que creen que va a pasar en este caso?

```
1  int main(void) {
2    while(1){
3      fork();
4    }
5  }
```

A este caso se le llama **Fork Bomb**.

Ejercicio

Escribir un programa que construya un árbol de procesos que represente la siguiente genealogía:

Abraham es padre de Homero,
    Homero es padre de Bart
    Homero es padre de Lisa
    Homero es padre de Maggie.

Cada proceso debe imprimir por pantalla el nombre de la persona que representa.