Examen parcial de Sistemas Operativos 1

Evaluación parcial 1

8 de abril del 2022

La prueba parcial tiene una duración de 2 horas (18:00-20:00), aunque la prueba se puede hacer en 1h30 aproximadamente. El parcial tiene las siguientes condiciones y restricciones. A partir de la página siguiente se muestra el enunciado de la prueba.

- **El enunciado de la prueba se publicará en el campus virtual el mismo día de la prueba a las 8:00,** dónde el alumnado podrá prepararse la prueba con el material del campus virtual, Internet, etc.
- La prueba parcial se debe responder de forma **individual**.
- **Las respuestas de la prueba pueden ser realizadas tanto en castellano como en catalán**.
- **NO se podrá traer apuntes a clase**.
- **A cada una de las páginas entregadas se debe indicar el nombre completo y NIUB**.
- Se utilizarán **4 páginas como máximo** para responder las preguntas de la prueba parcial. Es importante comentar y/o razonar las respuestas a las preguntas. Es suficiente responder de forma breve a cada una de las preguntas (con 3 o 4 frases) para que se dé la respuesta como válida. No es necesario extenderse.
- A la hora de evaluar se valorará el hecho de que el estudiante haya entendido lo que se dé como respuesta. Se recomienda pues responder a las preguntas con información que el estudiante comprenda. Evitad copiar texto o código de Internet, por ejemplo, de cosas que no se entienden. El profesorado podrá requerir al estudiante que explique alguna de sus respuestas posteriormente a la prueba.
- En caso necesario, el profesorado puede proporcionar un justificante de asistencia a la prueba.
- **Cualquier intento de copia comportará la aplicación de la normativa académica general de la Universidad de Barcelona y el inicio de un proceso disciplinario.**

Problema 1 (6.25 puntos en total, 0.75 puntos por pregunta salvo en los apartados en los que se indica lo contrario)

A continuación se muestra un código en C que compila y funciona correctamente.

```c
int main(void)
{
  int ret1, ret2, fd[2];
  char *argv1[5] = {"/usr/bin/find", "gutenberg", "-type", "f", NULL};
  char *argv2[3] = {"/usr/bin/wc", "-l", NULL};

  pipe(fd);

  ret1 = fork();

  if (ret1 == 0) {
    printf("Going to execute %s\n", argv1[0]);
    dup2(fd[1],1);
    close(fd[0]);
    close(fd[1]);
    execv(argv1[0], argv1);
  }

  ret2 = fork();

  if (ret2 == 0) {
    printf("Going to execute %s\n", argv2[0]);
    dup2(fd[0],0);
    close(fd[0]);
    close(fd[1]);
    execv(argv2[0], argv2);
  }

  close(fd[0]);
  close(fd[1]);

  printf("Waiting...\n");
  waitpid(ret2, NULL, 0);
  printf("Finished\n");
}
```

Al ejecutar el código anterior se muestran los siguientes mensajes por pantalla.

```
Waiting...
Going to execute /usr/bin/wc
Going to execute /usr/bin/find
2727
Finished
```

En los siguientes apartados se analizará paso por paso lo que hace el código. Se recomienda leer primero todo el enunciado antes de comenzar a responder las preguntas.

Para responder a las preguntas se pueden ignorar las llamadas a la función `close` que hay en el código. En caso que se prefiera no ignorarlos, indicarlo a la hora de responder. En caso de se haga algún otro supuesto indicarlo en la respuesta.

a) Comenta brevemente (dibujando en caso que haga falta) cuál es la estructura de padre-hijos que se genera al ejecutar el código.

b) Basándonos en la salida mostrada del código, ¿qué partes del código son las que ejecutan primero y cuáles después? Razona tu respuesta.

c) Comenta brevemente qué es lo que hace la línea 7 del código y dibuja el esquema de conexiones de comunicación asociado.

d) Comenta brevemente qué es lo que hace la línea 9 del código. ¿Qué posible(s) valor(es) devuelve esta llamada a sistema?

e) Comenta brevemente la diferencia entre la línea 11 y la línea 21. ¿Qué proceso(s) entrará(n) en cada uno de estos casos?

f) (0.5 puntos) ¿Qué es lo que hace la línea 13 del código? ¿Por qué se ejecuta esta línea? Es decir, ¿cuál es el objetivo de ejecutar esta línea?

g) (0.5 puntos) De forma similar, ¿qué es lo que hace la línea 23 del código? ¿Por qué se ejecuta esta línea? Es decir, ¿cuál es el objetivo de ejecutar esta línea?

h) En las líneas 16 y 26 se ejecutan dos procesos. Dadas las instrucciones ejecutadas anteriormente, ¿qué es lo que hace el código? Puedes apoyarte a responder la pregunta indicando cuál es el comando equivalente a ejecutar en el terminal.

i) ¿Qué es lo que realiza la línea 33? ¿Por qué es necesaria?

Problema 2 (3.75 puntos en total, 0.75 puntos por pregunta) Comentar las siguientes afirmaciones de los sistemas operativos. En caso de que la frase sea incorrecta, indicar cómo funciona realmente y razonar la respuesta poniendo ejemplos si se cree conveniente. En caso que sea correcta, comentar la razón por la cual crees que es correcta.

a) La IEEE (Institute of Electrical and Electronics Engineers) desarrolló el estándar POSIX para facilitar al núcleo de cualquier sistema UNIX el envío de excepciones a los procesos.

b) La llamada a sistema `pipe` permite a los sistemas operativos UNIX realizar operaciones de comunicación entre procesos padre, hijos y nietos (un nieto es el hijo creado por un hijo).

c) Los procesos ejecutados por un usuario sin derechos de administrador (pej, oslab) de un sistema UNIX no podrán ejecutar instrucciones en modo núcleo al realizar una llamada a sistema.

d) Las interrupciones son señales síncronas utilizadas por el sistema operativo para tomar el control del mismo.

e) En un sistema con una única CPU, dos procesos pueden estar ejecutándose en un instante determinado, cada uno utilizando el 50 % de la CPU.