Universidad Nacional de Ingeniería
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación
Sistemas Operativos Avanzados (CC-571)
Ciclo: 2020-I

Primera práctica calificada

Normas

- Presentar cada una de sus respuestas en un archivo separado. Los archivos separados tendran el nombre de solución1.c o solución2.mkd si es que es la respuesta es texto.
- Las respuestas de texto deben tener más 200 palabras. No se aceptan respuestas de menos palabras y se tomará en cuenta la ortografía y la semántica de las soluciones.
- La entrega es en un archivo comprimido.
- No se admiten copias. **Cualquier evidencia de copia de otra fuente no serán consideradas y el puntaje ser cero**.

Preguntas

1. Responde las siguientes preguntas:

   - ¿Qué es un registro, pila, heap, puntero de instrucciones, puntero de pila?
   - ¿Qué proporciona el tiempo compartido a la virtualización?
   - ¿Debería haber un límite en la cantidad de procesos que un usuario puede ejecutar simultáneamente?
   - ¿Por qué es una buena idea separar el mecanismo de las políticas?
   - Muestra el PCB de Linux
   - ¿Cómo dificulta la ejecución directa que el OS cambie entre procesos?. ¿Cómo resuelve el OS este problema?
   - ¿Cómo la ejecución directa limitada permite que el OS virtualice la CPU?
   - ¿Qué hace el planificador (en términos simples)?
   - ¿Cómo mejora SJF sobre FIFO?
   - ¿Cómo mejora STCF sobre SJF?.

   Nota: Debes presentar figuras y código fuente en C o python para justificar tus respuestas .

2. Desarrollo los siguientes (de preferencias escribir código funcional para sus respuestas):

   - Considere el siguiente código de un programa de shell.

```
comando = leer_usuario();
int rc = fork();
if(rc == 0) { //proceso hijo
   exec(command);
}
else {// proceso padre
  wait();
}
```

Ahora, supongamos que el shell desea redirigir la salida del comando no a `STDOUT` sino a un archivo `f.txt`. Muestra cómo modificarías el código anterior para lograr esta redirección de salida.

- Considera un proceso padre P que ha bifurcado un proceso hijo C en el siguiente programa a continuación

```
int a = 5;
int fd = open(...) //abriendo un archivo
int ret = fork();
if(ret >0){
   close(fd);
   a = 6;
   ...
  }
 else if(ret==0){
    printf("a=%d\n", a);
    read(fd, texto);
 }
```

Una vez que se bifurca el nuevo proceso, suponga que el proceso padre se programa primero, antes que el proceso hijo. Una vez que el padre se reanuda después de la bifurcación, cierra el descriptor de archivo y cambia el valor de una variable como se muestra arriba. Suponga que el proceso hijo se programa por primera vez solo después de que el padre complete estos dos cambios.

  - ¿Cuál es el valor de la variable a si se imprime en el proceso hijo, cuando se programa a continuación? Explica tu respuesta.
  - ¿El intento de leer del descriptor de archivos tendrá éxito en el proceso hijo? Explica tu respuesta.

3. Responde las preguntas:

- Considere los siguientes eventos que ocurren durante un cambio de contexto del (modo de usuario) proceso P al (modo de usuario) proceso Q, desencadenado por una interrupción del temporizador que ocurrió cuando P se estaba ejecutando, en un diseño de un OS tipo Unix. Organiza los eventos en orden cronológico, comenzando desde el el primero al último. Explica tu respuesta.
  - (A) El contador del programa de la CPU se mueve del espacio de direcciones del kernel de P al espacio de direcciones del kernel de Q.
  - (B) El proceso de ejecución de la CPU P se mueve del modo de usuario al modo de kernel.
  - (C) El puntero de la pila de la CPU se mueve de la pila kernel de P a la pila kernel de Q.
  - (D) El contador del programa de la CPU se mueve del espacio de direcciones del kernel de Q al espacio de direcciones del usuario de Q.

  - (E) Se invoca el código del planificador del OS.
- ¿Cuál de las siguientes piezas de información sobre un proceso son idénticas para un proceso padre y el proceso hijo recién creado, inmediatamente después de completar la llamada al sistema `fork`?:
  - El identificador del proceso.
  - El contenido de la tabla de descriptores de archivo.
    Explica tu respuesta.
- Cuando un proceso realiza una llamada al sistema y ejecuta el código del kernel:
  - ¿Cómo obtiene el proceso la dirección de la instrucción del kernel para saltar?
  - ¿Dónde se almacena el contexto del espacio de usuario del proceso (contador de programa y otros registros) durante la transición del modo de usuario al modo kernel?.

4. En este ejercicio, mediremos el costo de una llamada al sistema con dos archivos dados: `f1.c`, `f2.c` y `f3.c`.

   - Explica cada uno de los resultados y sus diferencias.
   - ¿Qué hace las sentencias: `clock_gettime,CLOCK_BOOTTIME CLOCK_REALTIME` en los dos primeros archivos.

     ¿Encuentra y explica algunas diferencias entre `clock_gettime`, `gettimeofday` mencionada en clase.