Sistemas Operativos

Departamento de Computación – FCEyN – UBA

Primer cuatrimestre de 2023

Ejercicios de repaso (2do parcial) – 27/06 - Primer cuatrimestre de 2023

Ejercicio 1. Drivers

Se requiere diseñar un driver para instalar en un robot transportador de paquetes para un servicio de correos. El mismo está compuesto por: una batería, una pequeña computadora que corre un sistema operativo Linux, un lector de código de barras, un brazo mecánico y un sistema de movimiento.

El modo de funcionamiento es el siguiente: el robot corre un programa de usuario que ejecuta la función `siguiente_paquete()` para obtener un dato del tipo `struct paquete(int x, int y, int codigo)` que representa la localización `(x,y)` de alguna de las estanterías del depósito, y el código de barras del paquete a transportar. Luego, deberá indicarle al sistema de movimiento que se desplace hasta dicha ubicación. El desplazamiento puede demorar una cantidad de minutos imposible de saber previamente, por lo que el sistema de desplazamiento deberá informar de algún modo cuando haya llegado a la ubicación correcta.

Cada posible ubicación representa una estantería que contiene 10 paquetes. El robot deberá buscar el paquete correcto mediante el siguiente procedimiento. Deberá tomar cada paquete uno por uno, utilizando para ello su brazo mecánico, y leer su código de barras. Si el código no coincide, se deberá volver a dejar el paquete en la estantería, y leer el siguiente paquete. Si el código coincide, se deberá llevar el paquete hacia la posición `(0,0)` del depósito, en donde se encuentra una cinta de entrega de paquetes. Finalmente, se deberá depositar el paquete en dicha cinta.

Cuando se encuentre frente a una estantería, el brazo mecánico puede extenderse y moverse de forma horizontal y vertical a cada una las 10 posiciones correspondientes de una estantería. Cuando se encuentre frente a la cinta, puede extenderse sobre la cinta. También puede contraerse para acercarse al lector de códigos de barras. En su punta cuenta con una mano que puede tomar y soltar cosas. Asumir que los movimientos del brazo son tan veloces que se pueden considerar instantáneos. El brazo deberá estar contraído para poder ser leído por el lector de códigos de barras.

Tener en cuenta en el diseño que la duración de la batería en el robot se ve seriamente afectada cuando el procesador es usado intensivamente. Asumir que los paquetes se reponen de forma automática en las estanterías (siempre hay 10 paquetes disponibles). Asumir que no existe una estantería en la posición `(0,0)`. Asumir que siempre se va a poder encontrar el paquete buscado.

a) Proponga un diseño, en donde debe indicar cuántos y qué tipo de registros tendría cada dispositivo, e indicando también para qué se utilizarían. También debe indicar y justificar el tipo de interacción con cada dispositivo (*interrupciones, polling, dma, etc.*). Puede usar uno o más tipos de interacción diferentes para cada dispositivo. Se debe considerar el uso de la batería para justificar las decisiones.

b) Una vez que tenga el diseño, debe escribir los tres *drivers* correspondientes a lector, brazo y sistema de movimiento respectivamente. Escribir con código C todas las operaciones que considere necesarias para poder cumplir el objetivo planteado: *init, load, open, write, read, etc.* Tenga en cuenta que el software de control correrá a nivel de usuario, y podrá tener interacción con los drivers. Indicar con código C cómo interactuará el software de control con los *drivers*. Cada operación usada debe estar justificada.

c) Explique cómo se genera la interacción a nivel del código entre los *drivers* que propuso.

Ejercicio 2. Seguridad

El siguiente fragmento de código en C presenta vulnerabilidades que permite lograr un **escalamiento de privilegios**. Se deberá realizar un análisis del mismo siguiendo las siguientes pautas:

- Explique *brevemente* qué hace el código, qué función cumple cada variable, qué condiciones deben existir para que se ejecute completamente, y cuales serían los vectores de ataque. Justifique e indique cualquier detalle relevante.
- Indique los nombres de las vulnerabilidades involucradas.
- Indique con qué argumentos y entradas de usuario deberá ser ejecutado el programa para vulnerarlo. En caso de requerir contar con un payload complejo, indicar la composición del mismo.
- En caso de requerir conocer algún dato o valor del programa para construir el payload o para realizar el ataque, indique de dónde proviene el mismo, y cómo lo obtendría.
- Indique y justifique las posibles formas de impacto del ataque en base a los tres principios básicos de la seguridad de la información.
- Proponga una forma de mitigar la(s) vulnerabilidad(es) modificando el código.
- Indique y justifique si existen formas de mitigar la(s) vulnerabilidad(es) a nivel del sistema operativo.

```
1  /**
2   * Dado un usuario y una clave, indica si la misma es válida
3   */
4  extern bool clave_es_valida(char* usuario, char* clave);
5
6  bool validar_clave_para_usuario(char *usuario){
7    // fmt = "%......."
8    char fmt[8];
9    fmt[0] = '%';
10
11   printf("Ingrese un largo para la clave: ");
12
13   // fmt = "%NNNN\0"
14   scanf("%4s", fmt+1);
15   int l = strlen(fmt+1);
16
17   // max_size <- atoi(NNNN)
18   unsigned char max_size = atoi(fmt+1);
19   char clave[max_size+1];
20
21   // fmt = "%NNNNs\0"
22   fmt[l+1] = 's';
23   fmt[l+2] = '\0';
24
25   scanf(fmt, clave);
26
27   return clave_es_valida(usuario, clave);
28 }
29
30 int main(int argc, char **argv){
31   setuid(0);
32   bool es_valida = validar_clave_para_usuario(argv[1]);
33   if(es_valida) {
34     system("/bin/bash");
35   } else {
36     exit(EXIT_FAILURE);
37   }
38 }
```

Ejercicio 3. Filesystems

(Ejercicio pendiente...)

Ejercicio 4. Distribuidos

(Ejercicio pendiente...)