Entrada/Salida

Sistemas Operativos

Franco Frizzo

9 de mayo de 2017

Enunciado

Una pequeña empresa de logística acaba de adquirir un robot que permite localizar y obtener cajas en su depósito.

Cuando se le ingresa un código en el registro de 32 bits `LOC_TARGET` y la constante `START` en el registro `LOC_CTRL`, el robot comienza la operación de búsqueda, escribiendo el valor `BUSY` en el registro `LOC_STATUS`.

Al encontrar la caja, la deposita en la bandeja de salida, escribe el valor `JOYA` en el registro `LOC_CTRL` y el valor `READY` en el registro `LOC_STATUS`. Si no puede encontrar la caja, escribe el valor `BAJON` en el registro `LOC_CTRL` y el valor `READY` en el registro `LOC_STATUS`. En todos los casos el contenido de `LOC_TARGET` se mantiene hasta tanto se vuelva a escribir otro valor.

El robot vino con el siguiente software:

```
1  int main (int argc, char *argv[]) {
2      int robot = open("/dev/chinbot", "w");
3      int codigo;
4      int resultado;
5      while (1) {
6          printf("Ingrese el código de la caja\n");
7          scanf ("%d", &codigo);
8          resultado = write(robot, codigo);
9          if (resultado == 1) {
10             printf("Su orden ha llegado\n");
11         } else {
12             printf("No podemos encontrar su caja %d\n", codigo);
13         }
14     }
15 }
```

Desafortunadamente, el *driver* que vino con el robot parece no ser compatible con el sistema operativo que utiliza la empresa. Al intentar comunicarse con los fabricantes para obtener soporte, la respuesta que obtuvieron fue "谢谢。很快回来。". Por lo tanto, han decidido recurrir a nuestra ayuda.

1. Implementar la función `int driver_write(void* data)` del *driver*. Asegurarse de que el código sea reentrante. ¿Qué método de acceso emplea la solución?

2. Parece que el manual del robot, escrito en un dudoso castellano, contiene la siguiente información:

> "*Robot es compatible con el acceso de interrupción. Se selecciona este modo, una operación terminada `CHINBOT_INT` interrupción lanzará.*"

Aprovechando esta información, modificar el código del ítem anterior para que utilice interrupciones.

Resolución

1. *Polling*

```
1  mutex acceso;
2
3  int driver_init() {
4      acceso = mutex_create();
5  }
6
7  int driver_write(void* data) {
8      int codigo = copy_from_user(data);
9
10     acceso.lock();
11     OUT(LOC_TARGET, codigo);
12     OUT(LOC_CTRL, LOC_START);
13
14     while (IN(LOC_STATUS) != BUSY) {}
15     while (IN(LOC_STATUS) != READY) {}
16
17     resultado = IN(LOC_CTRL);
18     acceso.unlock();
19
20     if (resultado == JOYA) {
21         return 1;
22     }
23     else if (resultado == BAJON) {
24         return 0;
25     }
26     return -1;
27 }
```

2. Interrupciones

```
1  mutex acceso;
2  semaforo listo;
3  bool esperando;
4
5  int driver_init() {
6      acceso = mutex_create();
7      listo = semaforo_create(0);
8      esperando = false;
9      irq_register(CHINBOT_INT, handler);
10 }
11
12 int driver_write(void* data) {
13     int codigo = copy_from_user(data);
14
15     acceso.lock();
16     OUT(LOC_TARGET, codigo);
17     OUT(LOC_CTRL, START);
18
19     esperando = true;
20     listo.wait();
21
22     resultado = IN(LOC_CTRL);
23     acceso.unlock();
24
25     if (resultado == JOYA) {
26         return 1;
27     }
28     else if (resultado == BAJON) {
29         return 0;
30     }
31     return -1;
32
33 }
34
35 void handler() {
36     if (esperando && IN(LOC_STATUS) ==
       READY) {
37         esperando = false;
38         listo.signal();
39     }
40 }
```