Atividade 7. Desenvolvimento de Device Drivers

(Mácio Monteiro de Meneses Júnior ~mmmj,

Maria Isabel Fernandes dos Santos ~mifs)

1 - (0,75) Qual a relação entre minor number, major number, driver e dispositivo?

- O major number identifica qual é o driver que está associado a nomes no sistema de arquivos e minor number identifica o dispositivo entre os demais dispositivos associados ao mesmo driver

2 - (0,75) O que são device drivers e quais são as funcionalidades comuns que eles possuem?

- Driver é a camada de SW responsável pela comunicação com um dispositivo. O driver geralmente provê 6 funcionalidades básicas para utilizar o dispositivo:

  . Inicialização/Reset do dispositivo
  . Abertura
  . Fechamento
  . Leitura
  . Escrita
  . Controle de E/S

3 - (0,5) Diferentes as três classes de dispositivos definidas no Linux (char, block e network)

- Character devices: a comunicação é feita através de sequências de bytes, usada em monitores, teclados e mouses.
- Block devices: a comunicação é feita através de blocos de dados com capacidade de armazenamento (através de buffers), e é tipicamente utilizado para disco.
- Network devices: a comunicação feita através de pacotes.

4 - (1,0) Qual a relação da struct file_operations com as interfaces oferecidas pelo sistema operacional para as aplicações do usuário (ex. read, write, open, etc)?

- A struct file_operations tem papel importante no driver, pois contém os ponteiros para as funções que implementam o comportamento desse driver. Suas chamadas de sistema utilizam as funções apontadas pelo file_operations (open, read, write, etc).

5 - Sobre dispositivos e drivers de caracteres (Character devices e Character drivers)
a) Descreva o que é um dispositivo de caractere.

- Character devices possuem comunicação feita através de sequência de bytes. Como exemplo, podemos citar o mouse, o teclado e o monitor.

6 - (1,0) A figura abaixo ilustra os registradores de configuração PCI. Baseado neles, responda/faça
a) (0,25) Qual a finalidade dos registradores Base Address 0-5 ?

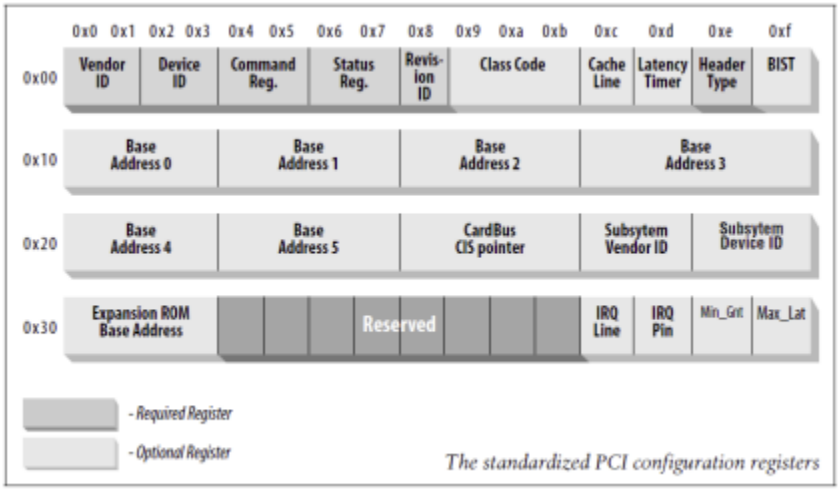
- Informar qual região de memória o dispositivo está mapeado, para ser possível ler e escrever nesse dipositivo.

b) (0,5) Descreva os campos vendorID, DeviceID e Class code.

- O vendorID e o deviceID identificam o fabricante e o dispositivo. O class code identifica o tipo de dispositivo (dispositivo de rede, de vídeo, etc).

c) (0,25) Quais campos podem ser utilizados para identificar um dispositivo PCI?

- O deviceID junto com o vendorID.



7 - (0,75) Qual é o tipo de struct fornecida pelo Linux utilizada no desenvolvimento de um driver PCI para elencar os dispositivos que o driver suporta? Dê um exemplo de uso.

- struct pci_device_id.

8 - (0,75) Um driver PCI deve criar uma estrutura do tipo struct pci_driver para poder se registrar no kernel do Linux. Quais os campos necessários que essa

estrutura possui? Dê um exemplo de uso

- const char* name; // nome do driver
  const struct pci_device_id *id_table; // Tabela de ids dos dispositivos ligados ao driver
  Função prove
  Função remove

  Ex:

  ```
  static struct pci_driver pci_driver = {
          .name = “Driver PCI Exemplo”,
          .id_table = pci_ids,
          .probe = pci_probe,
          .remove = pci_remove
  };
  ```

9 - (0,25) Cite uma função do kernel do Linux que pode ser adotada para registrar um driver PCI.

- pci_register_driver.

10 - (1,0) Quais as finalidades das funções de probe e remove no desenvolvimento de drivers PCI.

- As funções estabelecem ações de inicialização e finalização. A função probe é chamada pelo kernel assim que ele identifica algum dispositivo que possa utilizar o driver e aloca estruturas e mapeia as portas de E/S. A função remove é chamada pelo kernel quando dispositivo é removido e libera recursos alocados, desfaz o mapeamento das portas de E/S, etc.

11 - (1,0) Descreva os 4 tipos de endpoints USB (tipos de transferências)

- Os 4 tipos de endpoints são o control, um tipo de transferência usado para configurar o dispositivo; o interrupt, que é um tipo de transferência usado para transferir pequena quantidade de dados em um determinado período de tempo; o bulk, que é um tipo de transferência usado para a transferir grande quantidade de dados sem perda; Por fim, há o Isochronous, também um tipo de transferência usado para transferir grande quantidade de dados, podendo haver perda.

12 - (0,5) Descreva a relação entre endpoint, interface e driver USB.

- Um endpoint é um ponto endereçável de um dispositivo USB. Uma interface é um conjunto de endpoints. Para cada interface, é necessário um driver, e um dispositivo pode ter vários drivers.

13 - (0,5) O que é uma URB (USB Request Block) e qual seu ciclo de vida?

- É uma estrutura de dados que faz a comunicação com um determinado dispositivo. É utilizada para transferência de dados entre endpoints de um dispositivo. Seu ciclo de vida inicia-se com a criação pelo driver USB, atribuição a um endpoint específico de um dispositivo, submissão ao USB Core pelo driver USB, submissão ao controlador USB pelo USB Core, processamento pelo controlador USB e envio ao dispositivo, e por fim notificação ao driver pelo controlador quando o URB é atendido pelo dispositivo.

14 - (0,5) Quais os campos presentes na struct que define uma URB (struct urb) ?

- São 5 campos:
  . O usb_device em que vai ocorrer a transferência.
  . O endpoint do usb_device.
  . Um ponteiro para um buffer onde os dados serão lidos ou armazenados.
  . O tamanho do buffer.
  . Um ponteiro para a função que será chamada ao término da transferência de dados.

15 - (0,5). No desenvolvimento de drivers, quais as macros utilizadas para identificar as funções que devem ser executadas no carregamento e no descarregamento do driver no kernel do Linux?

- module_init e module_exit.