Edimilton Rocha Santana Ferreira

Accessibility Mouse Driver

Driver de Mouse USB Personalizado para Linux

Interface Hardware e Software
Bruno Otavio Piedade Prado

Tópicos

- Introdução
- Motivação
- Planejamento
- Metodologia
- Customização do Módulo
- Demonstração
- Conclusão

Nome da Empresa

Introdução

No mundo digital em que vivemos, o uso de um mouse é fundamental para interagir com computadores e dispositivos. No entanto, essa tarefa pode ser desafiadora para pessoas com deficiência (PCD), especialmente aquelas com dificuldades motoras.

Motivação

A motivação por trás do projeto é abordar essa lacuna na acessibilidade. Reconhecemos a importância de garantir que a tecnologia seja verdadeiramente inclusiva, e aprimorar a experiência do usuário para PCD é uma parte fundamental desse esforço.

O driver de mouse USB personalizado oferece uma solução elegante e acessível para esse problema. Ele permite um controle simplificado do cursor, facilitando a navegação em toda a tela com pequenos movimentos do mouse. Além disso, oferecemos uma maneira fácil de ativar e desativar esse recurso, tornando-o flexível e adaptável às necessidades individuais dos usuários.

Planejamento

Semana 1

Estudei a documentação do kernel do Linux para compreender os princípios de desenvolvimento de módulos de kernel e as interfaces de dispositivos de entrada. Pesquisei fontes adicionais para criar um módulo de mouse personalizado.

Semana 2

Trabalhei na personalização do módulo usbmouse disponível no GitHub oficial do Linux. Isso incluiu a análise do código-fonte existente, identificação das áreas que precisavam de alterações e implementação das modificações necessárias.

Semana 3

Implementei as alterações n para permitir que o mouse navegue com base no último movimento detectado. Isso envolveu a criação de threads para lidar com a lógica de rastreamento de movimento e a capacidade de ligar e desligar essa funcionalidade com um botão lateral do mouse.

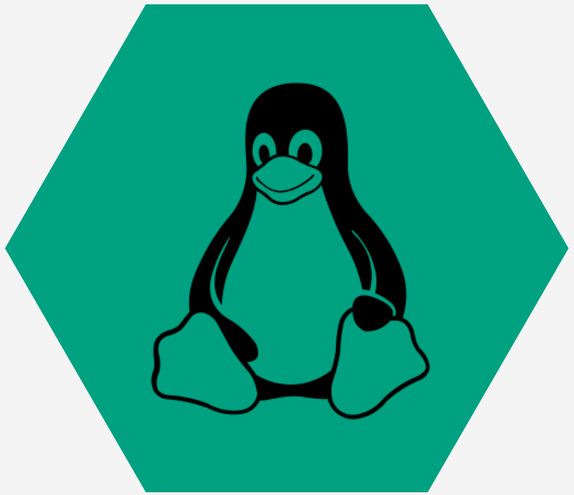
Atualmente

Realizei testes extensivos para garantir que as alterações funcionassem conforme o esperado e que não causassem problemas de estabilidade no sistema.

Metodologia

- Preparação do Ambiente de Desenvolvimento
- Estudo da Documentação do Kernel do Linux
- Pesquisa e Fontes de Informação
- Customização do Módulo USBMouse Padrão
- Testes e Validação



Customização do Módulo

```
// USB devices have different data layouts is required to configure the layout used by the mouse
input_report_key(dev, BTN_LEFT,   data[0] & 0x01);
input_report_key(dev, BTN_RIGHT,  data[0] & 0x02);
input_report_key(dev, BTN_MIDDLE, data[0] & 0x04);
input_report_key(dev, BTN_SIDE,   data[0] & 0x08);
input_report_key(dev, BTN_EXTRA,  data[0] & 0x10);

input_report_rel(dev, REL_X,     data[2]); // standard data[1]
input_report_rel(dev, REL_Y,     data[4]); // standard data[2]
input_report_rel(dev, REL_WHEEL, data[6]); // standard data[3]

if(data[2] != 0 || data[4] != 0) {
        current_x = data[2];
        current_y = data[4];
        previous_x = current_x;
        previous_y = current_y;
}

// Monitoring the status of the side button (BTN_SIDE)
side_button_state = data[0] & 0x08;
handle_side_button(dev, side_button_state);
```

- Trecho do código responsável por receber e salvar o valor do ultimo movimento detectado no mouse
- Trecho do código responsável por receber e salvar quando o botão lateral e pressionado.

```
static void handle_side_button(struct input_dev *dev, int value) {
    if (value) {
        movement_mode_active = !movement_mode_active; // Switching the status of the motion mode
    }
}

static int monitoring_kthread(void* in_dev) {
    struct input_dev *dev = (struct input_dev *)(in_dev);
    int speed_x;
    int speed_y;

    while (!kthread_should_stop()) {
        while(movement_mode_active) {
            speed_x = 8 * current_x;
            speed_y = 8 * current_y;
            input_report_rel(dev, REL_X, speed_x);
            input_report_rel(dev, REL_Y, speed_y);
            input_sync(dev);
            msleep(100);
        }
        schedule();
    }
    return 1;
}
```

- Trecho do código responsável por alternar entre os estados do mouse
- Trecho do código responsável por manter o movimento assincrono.

Conclusão

Em resumo, o Driver de Mouse USB personalizado para navegação simplificada é uma ideia que busca melhorar significativamente a acessibilidade e a experiência do usuário, especialmente para pessoas com deficiência.

Estou disponível para perguntas e discussões, e agradeço por terem assistido a apresentação até aqui.

Voltar ao slide de tópicos