

Projeto prático

Implementação da tela azul (Blue Screen of Death) em Linux

Isadora, Matheus Mortatti, Maurício Lorenzetti e Vitor Alves

professora

Islene Calciolari Garcia

MC504 - Sistemas Operacionais, Unicamp

14 de Junho, 2016

1 Ideia

- Objetivo
- Estratégias

2 Implementação

- Instanciar imagem na tela
- Impressão de texto

Objetivo

O objetivo inicial do projeto se baseava na implementação da **Blue Screen of Death**.

- 1 Usuário entra normalmente em seu sistema em Linux.

Objetivo

O objetivo inicial do projeto se baseava na implementação da **Blue Screen of Death**.

- 1 Usuário entra normalmente em seu sistema em Linux.
- 2 O sistema aciona um kernel panic, devido a alguma falha.

Objetivo

O objetivo inicial do projeto se baseava na implementação da **Blue Screen of Death**.

- 1 Usuário entra normalmente em seu sistema em Linux.
- 2 O sistema aciona um kernel panic, devido a alguma falha.
- 3 Uma tela azul é instanciada, com informações de DEBUG sobre as possíveis causas do kernel panic.

Exemplo

A problem has been detected and windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

Estratégias

Foram estabelecidas diferentes estratégias de soluções para a implementação do projeto.

- 1 Instanciar uma imagem na tela;
- 2 Imprimir o texto de DEBUG;
- 3 Adicionar features, como áudio, etc.

Instanciar imagem na tela

- Foram utilizados códigos de `driver/video` como referência para implementação;

Instanciar imagem na tela

- Foram utilizados códigos de driver/video como referência para implementação;
- Códigos voltados para graphics support;

Instanciar imagem na tela

- Foram utilizados códigos de `driver/video` como referência para implementação;
- Códigos voltados para `graphics support`;
- Exemplos instanciavam logos na tela a partir de imagens `.ppm`;

Instanciar imagem na tela

- Foram utilizados códigos de `driver/video` como referência para implementação;
- Códigos voltados para `graphics support`;
- Exemplos instanciavam logos na tela a partir de imagens `.ppm`;
- Problemas encontrados:
 - Instanciação de imagem na tela requer locks e permissões;
 - Dependente do tipo de display (VGA, HDMI...);
 - Permissões são restritas devido ao kernel panic;
 - Uma escrita arbitrária (não autorizada) no kernel leva ao caos!

Exemplo de imagem

- Imagem utilizada para renderização pelo driver:



Testes para impressão de texto

- Tentativas de impressão de texto a partir do acesso da região de memória do ponteiro de *Monochrome Text mode*, ie. 0xB0000.

Testes para impressão de texto

- Tentativas de impressão de texto a partir do acesso da região de memória do ponteiro de *Monochrome Text mode*, ie. 0xB0000.
- Entretanto, são necessárias permissões para acessar essa região de memória!
 - Kernel panic gerava outro kernel panic que gerava outro kernel panic e, por medida de segurança, o sistema reiniciava.

Testes para impressão de texto

- Tentativas de impressão de texto a partir do acesso da região de memória do ponteiro de *Monochrome Text mode*, ie. 0xB0000.
- Entretanto, são necessárias permissões para acessar essa região de memória!
 - Kernel panic gerava outro kernel panic que gerava outro kernel panic e, por medida de segurança, o sistema reiniciava.
- Investigação de métodos de `drivers/tty`, text-only console (**TeleTY**pewriter), para impressão na tela;

Testes para impressão de texto

- Tentativas de impressão de texto a partir do acesso da região de memória do ponteiro de *Monochrome Text mode*, ie. 0xB0000.
- Entretanto, são necessárias permissões para acessar essa região de memória!
 - Kernel panic gerava outro kernel panic que gerava outro kernel panic e, por medida de segurança, o sistema reiniciava.
- Investigação de métodos de `drivers/tty`, text-only console (**TeleTY**pewriter), para impressão na tela;
- Encontrada função `printk`, funcional em kernel panic.

Solução com a função printk

- Problema não estava resolvido. Output precisa ser **colorido**.

Solução com a função printk

- Problema não estava resolvido. Output precisa ser **colorido**.
- Função printk não suporta customização de cores, como a função printf.

Arquivo vt.c

- Lida com output no `tty`, text console.

Arquivo vt.c

















- Lida com output no tty, text console.
- Possui poucas funções visíveis para códigos do kernel.
 - Implementada nova função e adicionada no arquivo `include/vt.h`.

```
extern void vc_set_color(int* t_r);
```

Arquivo vt.c

```
/* *  
 * Helper function! Reset colors on the current VC  
 *   t_r: set as total number of rows of current vc  
 * */  
void vc_set_color(int* t_r)  
{  
    struct vc_data* vc = vc_cons[fg_console].d;  
  
    vc->vc_color = blue_screen;  
  
    update_attr(vc);  
  
    *t_r = vc->vc_rows;  
}
```

Máscara de cores

Number	Colour	Name	Number + bright bit	bright Colour	Name
0		Black	0+8=8		Dark Gray
1		Blue	1+8=9		Light Blue
2		Green	2+8=A		Light Green
3		Cyan	3+8=B		Light Cyan
4		Red	4+8=C		Light Red
5		Magenta	5+8=D		Light Magenta
6		Brown	6+8=E		Yellow
7		Light Gray	7+8=F		White

BSOD.c

- 1 Kernel panic ocorre e chama a função void `display_error()`, do arquivo `BSOD.c`;

BSOD.c

- 1 Kernel panic ocorre e chama a função `void display_error()`, do arquivo `BSOD.c`;
- 2 Função da mudança de cor do TTY é chamada;

BSOD.c

- 1 Kernel panic ocorre e chama a função `void display_error()`, do arquivo `BSOD.c`;
- 2 Função da mudança de cor do TTY é chamada;
- 3 Ambiente é modificado e `printk` é chamado, com mensagem a ser impressa.

Demonstração

```
echo c > /proc/sysrq-trigger
```