The use of events in BIOS practice

原创 Anthony
Posted on 2022-02-11 15:29:57 Read 1.5k Collection 1 Likes
Category Column: BIOS learning practice Article Tags: UEFI C

Here we mainly introduce two examples. The first one is seen in the previous Phoenix code, and the second one is encountered in the domestic platform. Let's look at the first one first.

Question: After the logo is displayed at startup, the logo still exists when entering the shell or pxe. It is easy to solve when entering pxe. We can just clear the screen before entering. So how to solve the problem after entering the shell?

Here we can use an event (why not clear the screen before entering the shell? I haven't practiced this, and I don't know whether the original solution has been verified, because compared with events, clearing the screen directly is simpler and more effective). After reading the simple use of events in UEFI principles and programming practice, here is a new knowledge point: the use of RegisterProtocolNotify

```
STATIC VOID  *mShellEnvProtocolCallbackReg = NULL;
STATIC EFI_GUID  gEfiShellEnvironment2Guid = {0x47c7b221, 0xc42a, 0x11d2, {0x8e, 0x57, 0x00, 0xa0, 0xc9, 0x69, 0x72, 0x3b}};
```

```
VOID
EFIAPI
ShellEnvProtocolCallback (
  IN EFI_EVENT                          Event,
  IN VOID                               *Context
  )
{
  gST->ConOut->ClearScreen(gST->ConOut);
}
```

```
EFI_EVENT                         ShellImageEvent;

if (mShellEnvProtocolCallbackReg == NULL) {
    Status = gBS->CreateEvent (
                    EVT_NOTIFY_SIGNAL,
                    TPL_CALLBACK,
                    ShellEnvProtocolCallback,
                    NULL,
                    &ShellImageEvent
                    );
    if (!EFI_ERROR (Status)) {
      Status = gBS->RegisterProtocolNotify (
                      &gEfiShellEnvironment2Guid,
                      ShellImageEvent,
                      &mShellEnvProtocolCallbackReg
                      );
    }
  }
```

收起 ^

The logic of the above code: a new event is added, and the response function of this event is a screen clearing function. So how to enter this event? You need to first understand the role of the RegisterProtocolNotify function, which is to determine the moment a certain protocol is installed, and then run a certain event. That is to say, after the shell installs gEfiShellEnvironment2Guid, this event takes effect, and the screen clearing operation is performed, successfully solving the above-mentioned problem.

Problem 2: When the UEFI display resolution is set to 1024x768, the Kylin system displays through VGA, which will cause split screen

Sometimes, a higher resolution is of course the best. For example, when the logo is displayed when the computer is turned on, looking at the blurry logo image always makes people doubt the performance of the machine. With a higher resolution, the resolution of the logo will also increase. This is definitely a good experience for users, but a good experience cannot always be achieved at the expense of normal use, so this problem must be solved. How to solve it? It is to solve it through events.

```
Status = gBS->CreateEventEx (
                 EVT_NOTIFY_SIGNAL,
                 TPL_NOTIFY,
                 ExitBootService,
                 NULL,
                 &gEfiEventExitBootServicesGuid,
                 &Event
                 );
```

AI generated projects 登录复制

```
VOID
ExitBootService (
  EFI_EVENT  Event,
  VOID       *Context
  )
{
  EFI_STATUS                      Status;
  UINT16                          OptionNumber;
  UINTN                           VarSize;
  CHAR16                          OptionName[9] = L"Boot";
  EFI_BOOT_MANAGER_LOAD_OPTION    BootOption;
  EFI_SIMPLE_FILE_SYSTEM_PROTOCOL *Volume = NULL;
  EFI_TPL  Tpl;

  Tpl = gBS->RaiseTPL (TPL_HIGH_LEVEL);
  gBS->RestoreTPL (TPL_APPLICATION);

  VarSize = sizeof (UINT16);
  Status = gRT->GetVariable (
                  L"BootCurrent",
                  &gEfiGlobalVariableGuid,
                  NULL,
                  &VarSize,
                  &OptionNumber
                  );
  if(EFI_ERROR(Status)) {
    return;
  }
  UnicodeSPrint (
    OptionName, sizeof (OptionName), L"%s%04x",
    OptionName, OptionNumber
    );
  Status = EfiBootManagerVariableToLoadOption (OptionName, &BootOption);
  if(EFI_ERROR(Status)) {
    return;
  }

  Volume = GetFsVolume (gBS, BootOption.FilePath, L"\\grub\\grub_ba.efi");
  if (Volume != NULL || !IsUefiAhciHddDp (gBS, BootOption.FilePath, NULL)) {
    // patch for kylin OS
    ResetGopDriver ();
  }

  gBS->RaiseTPL (Tpl);
}
```

收起 ︿

```
void
ResetGopDriver ()
{
  EFI_HANDLE          *HandleBuffer;
  UINTN               Index;
  UINTN               HandleCount;
  EFI_STATUS          Status;
  EFI_PCI_IO_PROTOCOL *PciIo;
  UINT32              Data32;
  BOOLEAN             IsAmdVga = FALSE;

  DEBUG ((EFI_D_INFO,"%a()\n",__FUNCTION__));

  Status = gBS->LocateHandleBuffer (
                  ByProtocol,
                  &gEfiPciIoProtocolGuid,
                  NULL,
                  &HandleCount,
                  &HandleBuffer
                  );
    if (!EFI_ERROR (Status)) {
    for (Index = 0; Index < HandleCount; Index++) {
      Status = gBS->HandleProtocol (
                      HandleBuffer[Index],
                      &gEfiPciIoProtocolGuid,
                      (VOID **)&PciIo
                      );
      if (!EFI_ERROR (Status)) {
        Status = PciIo->Pci.Read (PciIo, EfiPciIoWidthUint32, 8, 1, &Data32);
        Data32 = Data32 & 0xFFFFFF00;
        if (Data32 == 0x3000000 || Data32 == 0x00010000) {
          PciIo->Pci.Read (PciIo, EfiPciIoWidthUint32, 0, 1, &Data32);
          Data32 = Data32 & 0xFFff;
          if (Data32 == 0x1002) {
            IsAmdVga = TRUE;
            break;
          }
        }
      }
    }
  }

  if (IsAmdVga == FALSE) {
    if (HandleBuffer != NULL) {
      FreePool (HandleBuffer);
    }
    return;
  }

  PcdSet32 (PcdVideoHorizontalResolution, 640);
  PcdSet32 (PcdVideoVerticalResolution, 480);

  gBS->DisconnectController (HandleBuffer[Index], NULL, NULL);
  gBS->ConnectController (HandleBuffer[Index], NULL, NULL, TRUE);
  if (HandleBuffer != NULL) {
    FreePool (HandleBuffer);
  }

}
```

收起 ^