Universidade Federal de Uberlândia
FEELT – Faculdade de Engenharia Elétrica

Sistemas Embarcados II

Trabalho Semana 03

Professor: Éder Alves de Moura

Engenharia de Controle e Automação

Neila Cristina Morais Silva 12011EAU015

10/01/2022

Vídeo 01 - Bash in 100 Seconds

O vídeo comenta sobre BASH, que é um interpretador para interagir com computador por linhas de comando. Além de ser chamado de shell, pois envolve o kernel do computador do sistema operacional, permitindo o usuário escrever em arquivos e acessar dados digitando comando simples. Todo processo é através de um prompt onde o usuário pode digitar um comando, que será interpretado pelo shell e executado pelo sistema operacional.

Também é conceituado sobre BASH também ser uma linguagem de programação, ou seja, tudo digitado com uma linha de comando pode ser substituído por um script.

Vídeo 02 - Why so many distros? The Weird History of Linux

O vídeo mostra a história do Linus Torvalds de quando ainda era um estudante de 21 anos e começou a fazer um sistema operacional open source sem maiores pretensões, contrapondo aos grandes sistemas operacionais da época que o maior objetivo era fazer dinheiro.

O vídeo volta um pouco no tempo e conta a história da criação do sistema operacional unix e a guerra de detenção de direitos dos códigos desse sistema operacional, e partir dessa "guerra", foram surgindo outros sistemas operacionais, como GNU.

O GNU foi desenvolvendo cada vez mais aplicações parecidas com o unix, mas o GNU não tinha uma peça chave, o kernel, utilizado para unir software e hardware. E foi por essa época que Linus estava desenvolvendo seu sistema operacioal open source por hobbie. E em 1992 o código do Linux foi liberado sobre licença pública geral GNU.

A partir desse ponto, vários programadores foram inserindo aplicações do GNU no linux, em um SO que parecia o unix, mas que não poderiam ser processados.

Distros são baseadas no kernel do Linux e possuem vários pacotes e bibliotecas e geralmente um gerenciador de arquivos para instalar aplicações adicionais. Nos dias de hoje existem milhares de distros do Linux e cada um foi feita para servir a um propósito diferente. Distros para usos corporativos, usos domésticos, outras só podem ser usadas em servidores, mobiles e muito mais.

O restante do vídeo trata de mencionar sobre as diversas distros que surgiram para cada ramo citado a cima, coporativo, dméstico, segurança e diversos outros.

Em 2005 o código do linux era gerenciado por uma empresa privada, então

Linus Torvalds seguidor de sua filosia que era, criou o git, software gratuito para gerenciamento de códigos.

Vídeo 03 - 7 Linux Things You Say WRONG
O vídeo mostra algumas curiosidades sobre pronuncias erradas de nomes da comunidade.
Como errado (certo), GNU (Guh-New), Ubuntu (Uboontu), Dee-bian (Deb-ian), Mate (Mah-tay), openSUSE (Sue-sa) e comando SUDO (Sue-due).

Vídeo 04 - SLE15 - Introdução ao Linux Embarcado - Igor Tavares
O palestrante reforça a história do Linus Torvards e a criação do Linux, que já foram citadas em outros tópicos deste relatório.
A história dos sistemas embarcados e o linux se encontram nos anos 1990 em que as empresas que desenvolviam estavam crescendo bastante e cada vez lucrando mais, com produtos mais genéricos. E sair na frente da concorrência por menor tempo que fosse, era muito vantajoso no mercado, então tentando agilizar o time-to-market resolveram pegar o linux, bastante usado em outras aplicações e embarcar.
É mostrado alguns requisitos necessários para o linux embarcado, como as mínimas características de hardware, de cpu, memória ram e armazenamento. Também é falado sobre a necessidade de uma toolchain, e os componentes dessa toolchain, como compilador, debugger, assembler e linker, Standard C Library e headers do kernel. Outro item mencionado é o bootloader do linux, o kernel também é mostrado como o coração do sistema operacional e é citado suas funções. E por fim outros itens como, chamada de sistema e o sistema de arquivo e o build system.
As aplicações são citadas como uma grande ajuda para os desenvolvedores, e como é só as adaptar para o produto em questão que é possivel acelerar muito o tempo de prototipação.
Algumas placas de desenvolvimentos são mecionadas, como: edison, raspberry, odroid, beaglebone black entre outras.
O desenvolvedor de sistemas embarcados precisa de algumas habilidades, como conhecimentos em administração de sistemas linux, conhecimento de desenvolvimento de software, conhecimento de desenvolvimento de firmware, conhecimento de redes de computadores e conhecimento em projetos de hardware.

Livro Advanced Linux Programming
O livro descreve e conceitua-se o que é chamado de processo.
Que é uma instância que está rodando de algum programa, e que muitas vezes esses processos são usados para melhorar a performance de um programa, usando mais de um processo por vez.
3.1
ID de processo é um número que faz referência à cada processo do linux

unicamente, podemos assim diferencia os processos por esse ID, chamado de PID.
Cada processo é iniciado por outro processo (exceto o processo de init()), o que podemos concluir que cada processo possui o seu ID e é ligado a outro processo que o iniciou que possui outro ID, e para o processo iniciado o ID do processo que o abriu pode ser chamado de PPID, parent process ID.

```c
#include <stdio.h>
#include <unistd.h>


int main()
{
    printf("The process ID is %d\n", (int) getpid());
    printf("The parent process ID is %d\n", (int) getppid());
    return 0;
}
```

```
printingtheprocessID.c
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ gcc -o sprintingtheprocessID printingtheprocessID.c
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ ./sprintingtheprocessID
The process ID is 2212
The parent process ID is 1522
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$
```

3.1.2

```
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ ps
    PID TTY          TIME CMD
   1522 pts/0    00:00:00 bash
   2224 pts/0    00:00:00 ps
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ ps -e -o pid,ppid,command
    PID    PPID COMMAND
      1       0 /sbin/init splash
      2       0 [kthreadd]
      3       2 [rcu_gp]
      4       2 [rcu_par_gp]
      6       2 [kworker/0:0H-events_highpri]
      9       2 [mm_percpu_wq]
     10       2 [rcu_tasks_rude_]
     11       2 [rcu_tasks_trace]
     12       2 [ksoftirqd/0]
     13       2 [rcu_sched]
     14       2 [migration/0]
     15       2 [idle_inject/0]
     16       2 [cpuhp/0]
     17       2 [cpuhp/1]
     18       2 [idle_inject/1]
     19       2 [migration/1]
     20       2 [ksoftirqd/1]
     22       2 [kworker/1:0H-events_highpri]
     23       2 [kdevtmpfs]
     24       2 [netns]
     25       2 [inet_frag_wq]
     26       2 [kauditd]
     27       2 [khungtaskd]
     28       2 [oom_reaper]
     29       2 [writeback]
     30       2 [kcompactd0]
     31       2 [ksmd]
     32       2 [khugepaged]
     79       2 [kintegrityd]
     80       2 [kblockd]
     81       2 [blkcg_punt_bio]
     82       2 [tpm_dev_wq]
     83       2 [ata_sff]
     84       2 [md]
     85       2 [edac-poller]
     86       2 [devfreq_wq]
```

```
   1517       1 /usr/bin/xfce4-terminal
   1522    1517 bash
   2074       2 [kworker/1:0-events]
   2130       2 [kworker/0:0-events]
   2160       2 [kworker/u4:0-events_unbound]
   2189       1 mousepad /home/neila/code/SEII-NeilaCristinaMoraisSilva/Semana03/printingtheprocessID.c
   2195     908 /usr/libexec/dconf-service
   2200       2 [kworker/u4:1-ext4-rsv-conversion]
   2219       2 [kworker/u4:2-events_power_efficient]
   2220       2 [kworker/u4:3-events_unbound]
   2228    1522 ps -e -o pid,ppid,command
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$
```

3.2

```c
#include <stdlib.h>

int main()
{
    int return_value;
    return_value = system("ls -l /");
    return return_value;
}
```

```
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ gcc -o susingthesystemcall usingthesystemcall.c
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ ./susingthesystemcall
total 84
lrwxrwxrwx   1 root root     7 dez  7 21:16 bin -> usr/bin
drwxr-xr-x   3 root root  4096 jan  6 16:29 boot
drwxrwxr-x   2 root root  4096 dez  7 21:24 cdrom
drwxr-xr-x  20 root root  4120 jan  9 14:26 dev
drwxr-xr-x 135 root root 12288 jan  6 16:26 etc
drwxr-xr-x   4 root root  4096 dez 12 22:34 home
lrwxrwxrwx   1 root root     7 dez  7 21:16 lib -> usr/lib
lrwxrwxrwx   1 root root     9 dez  7 21:16 lib32 -> usr/lib32
lrwxrwxrwx   1 root root     9 dez  7 21:16 lib64 -> usr/lib64
lrwxrwxrwx   1 root root    10 dez  7 21:16 libx32 -> usr/libx32
drwx------   2 root root 16384 dez  7 21:16 lost+found
drwxr-xr-x   3 root root  4096 dez  7 21:45 media
drwxr-xr-x   2 root root  4096 ago 19 07:29 mnt
drwxr-xr-x   3 root root  4096 dez  7 21:50 opt
dr-xr-xr-x 231 root root     0 jan  9 14:25 proc
drwx------   3 root root  4096 ago 19 07:43 root
drwxr-xr-x  26 root root   800 jan  9 16:03 run
lrwxrwxrwx   1 root root     8 dez  7 21:16 sbin -> usr/sbin
drwxr-xr-x   2 root root  4096 dez  7 21:43 snap
drwxr-xr-x   2 root root  4096 ago 19 07:29 srv
dr-xr-xr-x  13 root root     0 jan  9 14:25 sys
drwxrwxrwt  14 root root 12288 jan  9 18:38 tmp
drwxr-xr-x  14 root root  4096 ago 19 07:32 usr
drwxr-xr-x  14 root root  4096 ago 19 07:43 var
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$
```

3.2.2

```c
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>

int main()
{
    pid_t child_pid;

    printf("the main program process ID is %d\n", (int) getpid());

    child_pid == fork();

    if (child_pid != 0){
        printf("this is the parent process with ID %d\n", (int) getpid());
        printf("the child's process ID is %d\n", (int) child_pid);
    }
    else
    printf("this is the child process with ID %d\n", (int) getpid());

    return 0;
}
```

```
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ gcc -o susingfork usingfork.c
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ ./ susingfork
bash: ./: Is a directory
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ ./susingfork
the main program process ID is 2329
this is the child process with ID 2329
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ this is the child process with ID 2330
```

3.3 Signals

Os sinais são usados para comunicar e manipular processos dentro do linux.

```c
#include<signal.h>
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<unistd.h>

sig_atomic_t sigusr1_count = 0;

void handler (int signal_number)
{
    ++sigusr1_count;
}

int main()
{
    struct sigaction sa;
    memset(&sa, 0, sizeof(sa));
    sa.sa_handler = &handler;
    sigaction(SIGUSR1, &sa, NULL);

    printf("SIGUSR1 was raised %d times\n", sigusr1_count);
    return 0;
}
```

```
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ gcc -o signalhandler signalhandler.c
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ ./signalhandler
SIGUSR1 was raised 0 times
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$
```

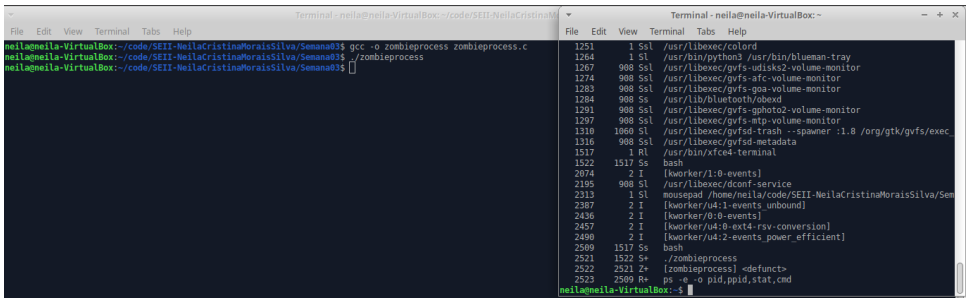
3.4 Process Termination

```
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ ls
printingtheprocessID.c  signalhandler  signalhandler.c  sprintingtheprocessID  susingfork  susingthesystemcall  usingfork.c  usingthesystemcall.c
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ ls neila
ls: cannot access 'neila': No such file or directory
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$ echo $?
2
neila@neila-VirtualBox:~/code/SEII-NeilaCristinaMoraisSilva/Semana03$
```

3.4.2

```c
#include<signal.h>
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<unistd.h>


int main()
{
    int child_status;

    char* arg_list[]={
    "ls",
    "-l",
    "/",
    NULL
    };

    spawn("ls", arg_list);

    wait(&child_status);
    if(WIFEXITED (child_status))
        printf("the child process exited normally, with exit code %d\n", WEXITSTATUS (child_status));
    else
        printf("the child process exited abnormally\n");

    return 0;
}
```

3.4.3



3.4.4

```c
#include<signal.h>
#include<string.h>
#include<sys/types.h>
#include<unistd.h>

sig_atomic_t sigusr1_count = 0;

void clean_up_child_process (int signal_number)
{
    int status;
    wait(&status);
    child_exit_status = status;
}


int main()
{
    struct sigaction sigchld_action;
    memset(&sigchld_action, 0, sizeof(sigchld_action));
    sigchld_action.sa_handler = &clean_up_child_process;
    sigaction(SIGCHLD, &sigchld_action, NULL);

    return 0;
}
```

4 - Linux kernel Development

É comentado sobre a criação e linha do tempo do linux e seu criador Linus Torvalds.

O linux é baseado no design monolithic, ou seja, tudo processa em um mesmo endereço, e o linux apesar de basear em várias coisas do unix, te sua própria independência, e quem decide as suas tomadas de decisões são os colaboradores da comunidade e o Linus Torvalds que controla o núcleo, por isso ao processar seus processos e programas o linux não prioriza nenhum tipo de atividade, a única coisa que o linux garante é que tudo será processado em algum tempo.

O espaço de usuário é aquele dinamicamente reservado para processar aplicações, enquanto o espaço do kernel são os dinamicamente reservados pra atividades do kernel.

Como na maioria dos kernels que funcionam bem, o kernel do linux são programados em c, e a biblioteca mais usada para compilação é a gcc da GNU.

A memória é protegida do espaço do usuário, então se uma região de memória protegida é chamada o kernel pode matar o processo. Mas se essa chamada de região protegida vir do kernel, isso pode causar alguns erros.

Não pode-se usar ponto-flutuante dentro do espaço, ou seja, da região de memória kernel.

O espaço do usuário possui uma grande e dinâmica pilha ao contrário do espaço do kernel que possui uma pilha pequena e fixa.

Um processo é uma instancia de um programa, um programa nãoé um processo, mas sim um parte desse programa que está utilizando de recursos, como mmemória RAM, hard disk, processamento, isso sim é um processo.

Um processo no linux é chamado pela função fork(), que copia o processo pai que chamou outro processo, enquanto o processo pai se mantém o mesmo, o processo filho executa os seus códigos.

Todo processo tem um número pid, é um número sequencial que vai surgindo com novos processos.

Thread é basicamente a menor parte de um processamento que pode ser processada em um sistema operacional.

Quando precisamos fazer várias tarefas dentro do mesmo processo, aí criamos várias threads e aí elas são executadas simultaneamente.

Para o kernel do linux não existe o conceito de threads, cada thread é considerada como um processo separado, mas podem ser executadas em concorrência e em paralelismo, ou seja, respectivamente, concorrendo pelos recursos da máquina e dividindo os recursos ao mesmo tempo.

O linux possui um subsistema para gerenciar processos, esse subsistema é

chamado de Process Scheduling.
Ele é responsável por dividir os recursos para os processos que estão sendo executados.
Ele faz parecer que vários processos estão sendo executados ao mesmo tempo, o que não é verdade, na verdade ele dá um pouco de recursos para o processo 01, depois o processo 01 vai dormir, ai o processo 02 recebe recursos, ai ele vai dormir também, e depois o gerenciador de processos da mais recursos para o processo 01, e assim segue.
O gerenciador de processos possui uma “policia“ que determina quanto tempo cada processo recebe.
Os processos são classificados de dois tipos, os que ficam esperando resposta de I/O e os que na maior parte do tempo ficam executando linhas de códigos, esses são os que levam prioridade, pois não vão enrolar esperando respostas de fora.
Timeslice é o tempo em que um processo será executado.
System call é um jeito programático em que um programa de computador requer um serviço do kernel, ou seja, do hardware .
São as bibliotecas c ou posix para essa comunicação, entre espaço do usuário e o sistema.
Cada System possui um system call number.