

# 07/10/2024 - Laboratorio de SO (Ude@)

## 1. Cronograma:

- Due time: Dead time:
- Entrega: 28/10/2024 04/11/2024
  - Sobre el curso de RHA: No se dejen colgar

## 2. Recursos para trabajar:

<https://udea-so.github.io/udea-so/docs/laboratorio/intro-laboratorio>

Curso de Sistemas Operativos Teoría Laboratorio Blog

Sobre el laboratorio

- Prácticas >
- Recursos >
- Tutoriales >
- intro
- Herramientas >
- Procesos
- Hilos y sincronización >

## Procesos

### 1. Llamados al sistema

Los procesos de usuario se ejecutan en el modo menos privilegiado de la máquina (modo usuario) donde sólo pueden ejecutar algunas operaciones básicas. Muchas de las operaciones privilegiadas relacionadas principalmente con los servicios del sistema operativo tienen que ser solicitadas al kernel mediante los **llamados al sistema**.

how to talk to your operating system

WRONG: good morning madam would you care to open a file for me

what? OPERATING SYSTEM

Tema: Laboratorio | Sistemas

Material de apoyo

Encuentros sincrónicos

Las grabaciones y los apuntes de los encuentros sincrónicos se encuentran en la siguiente lista [link]

Repositorio clases

Acceda al repositorio de los encuentros del laboratorio siguiendo el siguiente [link]

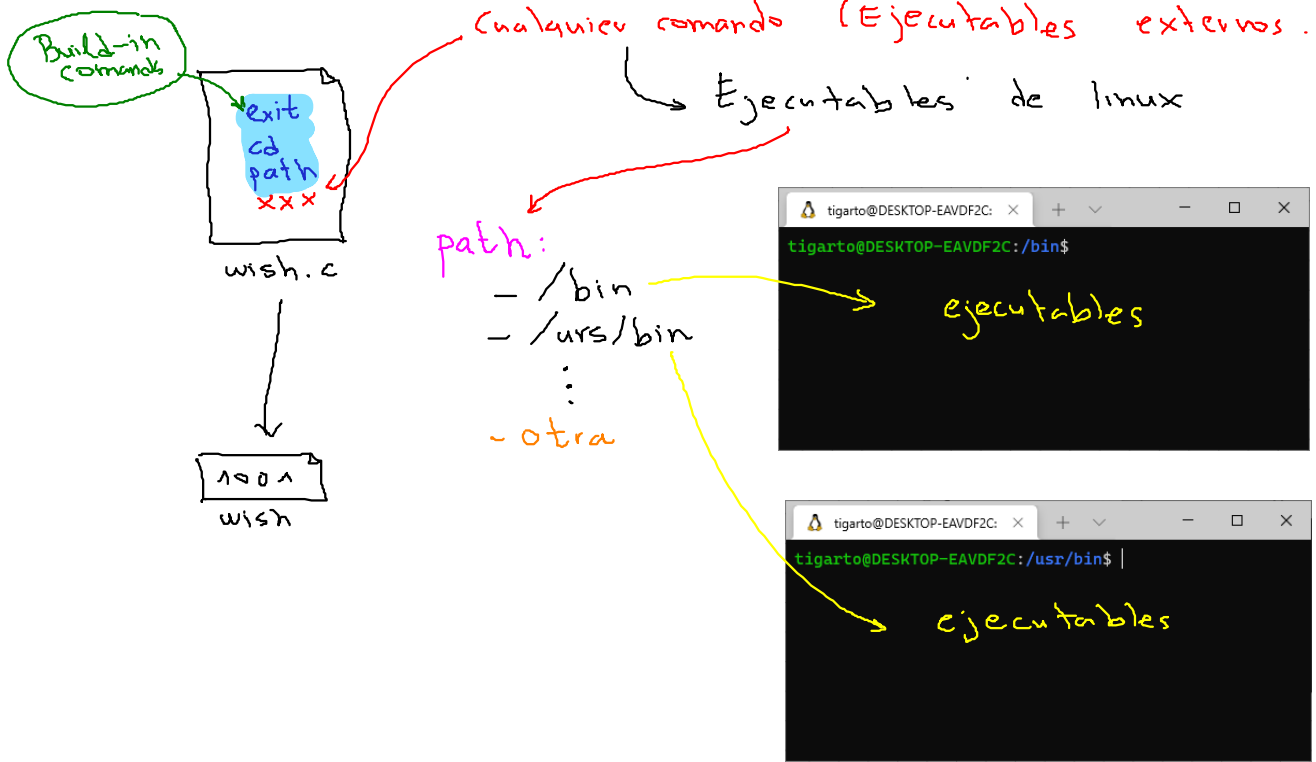
### Práctica 1

Encuentros sincrónicos

En la siguiente tabla se muestra la lista de grabaciones zoom del laboratorio.

Sesión	Grabación Zoom	Recursos
0	Sesión 0 (12/08/2024) [link]	[apuntes] - [agenda] - [repo]
1	Sesión 1 (26/08/2024) [link]	[apuntes] - [agenda] - [repo]
2	Sesión 2 (02/09/2024) [link]	[apuntes] - [agenda] - [repo]
3	Sesión 3 (02/09/2024) [link]	[apuntes] - [agenda] - [repo]
4	Sesión 4 (02/09/2024) [link]	[apuntes] - [agenda] - [repo]

### 3. Aspectos básicos

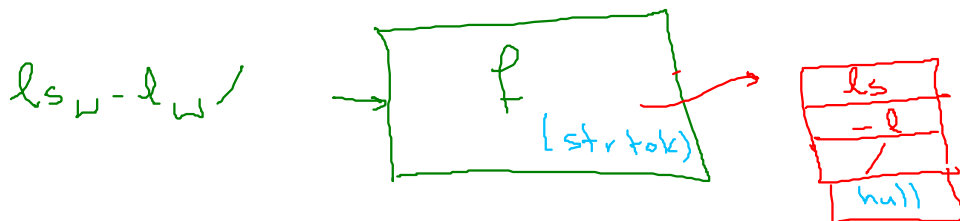


#### i. Modos de funcionamiento

- ↳ a. bash : Como los scripts. Ejemplo:  $\begin{cases} * sh \_ejemplo.sh \\ * python \_ejemplo.py \end{cases}$
- ↳ b. Interactivo : Se entran comandos por teclado

#### ii. Comandos:

↳ strsep  $\rightsquigarrow$  strtok ✓



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

```
char cmd[101]; // 100 [caracteres] + 1 [NULL]
```

```
int pid;
```

```
char path[] = "/bin/";
```

```
char full_cmd[151];
```

```
int cmd_tam;
```

```
int main(int argc, char *argv[]) {
```

```
if (argc != 1) {
```

```
    printf("Modo bash\n");
```

```
    // To Do...
```

```
    exit(0); // Exit OK
```

```
} else {
```

```
    // Modo terminal
```

```
    while (1) {
```

```
        printf("wish> ");
```

```
        fgets(cmd, 101, stdin);
```

```
        cmd_tam = strlen(cmd);
```

```
        cmd[cmd_tam - 1] = '\0'; // Sobreescribir enter
```

```
        // printf("%s %d %d\n", cmd, cmd_tam, strcmp(cmd, "hola"));
```

```
        if (strcmp(cmd, "hola") == 0) {
```

```
            // Comando interno (ejecutable hecho por el estudiante)
```

```
            pid = fork();
```

```
            if (pid == 0) {
```

```
                // Hijo ejecuta esto
```

```
                execl("./hola", "hola", NULL);
```

```
            } else {
```

```
                // Padre ejecuta esto
```

```
                wait(NULL); // Padre espera que el hijo acabe
```

```
            }
```

```
        } else if (strcmp(cmd, "exit") == 0) {
```

```
            // Comando interno (Se implementa como en el caso de hola)
```

```
            exit(0); // Exit OK
```

```
        }
```

```
        else if (strcmp(cmd, "cd") == 0) {
```

```
            // cd -> chdir()
```

```
            // Comando interno
```

```
            if (fork() == 0) {
```

```
                // Hijo
```

```
                execl("./cd", "cd", NULL);
```

```
            } else {
```

```
                wait(NULL); // Padre espera que el hijo acabe
```

```
            }
```

```
        }
```

```
    } else {
```

```
        // Comando externo (De los disponibles en linux - Puede ser el que sea)
```

```
        pid = fork();
```

```
        switch (pid) {
```

```
            case 0:
```

```
                // Hijo
```

```
                strcat(full_cmd, path);
```

```
                strcat(full_cmd, cmd);
```

```
                execl(full_cmd, cmd, NULL);
```

```
                break;
```

```
            default:
```

```
                // Padre
```

```
                wait(NULL); // Padre espera que el hijo acabe
```

```
                break;
```

```
            }
```

```
    }
```

```
}
```

```
}
```

```
// Vaciar el búfer para evitar problemas con el siguiente fgets
```

```
while (getchar() != '\n');
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```

Modos

Bash

Interactivo

stdin

Teclado

Comandos sin argumentos

ls

0 1 2 NULL

h	o	l	a	...
---	---	---	---	-----

3

## iii. Tipo de comandos:

→ Build-in Comandos (Internos)

```
else if(strcmp(cmd, "exit") == 0) {  
    // Comando interno  
    exit(0); // Exit OK  
}  
  
else if(strcmp(cmd, "cd") == 0) {  
    // Interno  
    // cd -> chdir() ← Todo...  
    printf("Uso del cd...\n");  
}
```

cd /home/xxx/...  
ruta

→ Externas

```
if(strcmp(cmd, "hola") == 0) {  
    // Comando interno (ejecutable hecho por el estudiante)  
    pid = fork();  
    if (pid == 0) {  
        // Hijo ejecuta esto  
        execl("./hola", "hola", NULL);  
    }  
    else {  
        // Padre ejecuta esto  
        wait(NULL); // Padre espera que el hijo acabe  
    }  
}
```

```
else {  
    // Comando externo (De los disponibles en linux - Puede ser el que sea)  
    pid = fork();  
    switch (pid) {  
    case 0:  
        // Hijo  
        strcat(full_cmd, path);  
        strcat(full_cmd, cmd);  
        execl(full_cmd, cmd, NULL);  
        break;  
    default:  
        // Padre  
        wait(NULL); // Padre espera que el hijo acabe  
        break;  
    }  
}
```

bin wish > ls  
full\_cmd  
✓ "/bin/"  
✓ ✓ "/bin/ls"  
exec

