دستورکار ۴ ــ آشنایی با چگونگی کارکرد فراخوان‌های سیستمی و پردازه‌های سیستم

زمان انجام: دو هفته

۱. **منظور از فضای کاربری (User Space) در سیستم‌عامل چیست؟**
۲. **منظور از مد کاربر (User Mode) و مد کرنل (Kernel Mode) چیست؟**
۳. **فراخوانی‌های سیستمی چگونه کار می‌کنند؟**
۴. فرایند init در لینوکس را بررسی کنید.
- **تلاش کنید این پردازه را بُکُشید. نتیجه‌ی تلاش‌ها و بررسی‌های خود را در گزارش کار بیاورید.**

۵. کارکرد فرایند fork را پیش‌تر بررسی کرده‌اید. پارامترهای ورودی و بازگشتی این فرایند را نیز بررسی کنید.
- **در سیستم عامل ویندوز و مک چه فرایندی کار فرایند fork را انجام می‌دهد؟**

۶. **نمونه‌ی کاربرد دستورهای pidof -s و pstree را بیاورید و توضیح دهید که چگونه کار می‌کند؟**
۷. **خروجی دستور زیر را بررسی کنید. نمونه‌ای از خروجی را در گزارش کار خود بیاورید و آن را توضیح دهید.**
- **ps -e -o pid,ppid,command**

۸. دو برنامه‌ی زیر را در لینوکس پیاده‌سازی، کامپایل و اجرا کنید.
- **تصویری از خروجی برنامه‌ها پس از اجرا روی دستگاه خود در گزارش کار بیاورید و آن را دقیقاً تحلیل کنید. کتابخانه‌های به‌کاررفته را نیز بررسی کنید.**
- **با کمک آنچه که از پرسش ۱۰ دستورکار پیشین آموخته‌اید ساختار پدر ــ فرزندی تا نخستین فرایند سیستم را برای فرایندهای برنامه‌ی شماره‌ی ۱ بنویسید.**

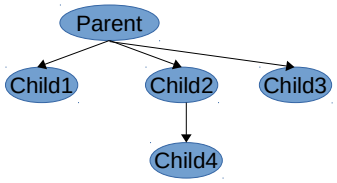
```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
        pid_t pid,ppid;
        pid = getpid();
        ppid = getppid();
        printf("\n1-%d\n",pid);
        printf("\n2-%d\n",ppid);
        fork();
        pid = getpid();
        ppid = getppid();
        printf("\n3-%d\n",pid);
        printf("\n4-%d\n",ppid);
        return 0;
}
```

برنامه‌ی ۱

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
        printf("Hello \n");
        fork();
        printf("bye\n");
        return 0;
}
```

برنامه‌ی ۲

۹. برنامه‌ای بنویسید که ساختار فرایندی به شکل زیر بسازد. نتیجه را در گزارش کار خود بیاورید و توضیح دهید. شماره‌ی فرایندها را در نموداری مانند زیر جایگزین کنید.



۱۰. از فراخوان سیستمی **execvp** در برنامه‌ای بهره ببرید به‌گونه‌ای که در فرایند فرزند یک دستور اجرا شود. مثلاً می‌توانید از دستور «**ls -l /**» یا «**ls | wc**» را به فراخوان **execvp** بدهید. برنامه را اجرا کنید و نتیجه‌های خروجی را در گزارش کار خود نشان دهید. کدام یک از فرایندهای پدر و فرزند اول اجرا می‌شود؟

۱۱. به‌کارگیری فراخوان سیستمی **wait**.

فراخوان سیستمی **wait** را به‌گونه‌ای به‌کار ببرید که فرایند پدر منتظر پایان اجرای فرایند فرزند بماند و سپس خودش اجرا شود. تصویر خروجی را در گزارش کار خود نشان دهید.

۱۲. برنامه‌ای بنویسید که در آن فرایندی را **fork** کند. در کد مربوط به فرایند پدر از فراخوان **sleep(60)** بهره ببرید تا فرایند پدر برای ۶۰ ثانیه به خواب برود. در کد مربوط به فرایند فرزند نیز با دستور **_exit(0)** برنامه را پایان دهید. در هنگام اجرای برنامه، فهرست فرایندهای در حال اجرای سیستم را با دستور **ps -e -o pid,ppid,stat,cmd** ببینید. (یک ترمینال دیگر باز کنید). حالت فرایند فرزند با چه چیزی نشان داده شده است؟ چرا؟ در این حالت اصطلاحاً گفته می‌شود فرایند فرزند زامبی (**Zombie**) شده است! چگونه می‌توان این زامبی را از سیستم بیرون کرد؟