دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

عنوان: تکلیف اول درس سیستم‌های عامل ۱

نام و نام خانوادگی: علیرضا ابره فروش

شماره دانشجویی: ۹۸۱۶۶۰۳

نیم سال تحصیلی: پاییز ۱۴۰۰

مدرّس: دکتر محمّدرضا حیدرپور

دستیاران آموزشی: مجید فرهادی - دانیال مهرآیین - محمّد نعیمی

فهرست مطالب

۱ عنوان سوال اول

اگر سوال بخش بندی شده نباشد، پاسخ آن در این قسمت نوشته می شود.

۱.۱ عنوان بخش اول سوال اول

پاسخ بخش اول سوال در این قسمت نوشته می شود.

۲.۱ عنوان بخش دوم سوال اول

پاسخ بخش دوم سوال در این قسمت نوشته می شود.

۳.۱ عنوان بخش سوم سوال اول

پاسخ بخش دوم سوال در این قسمت نوشته می شود.

۴.۱ عنوان بخش چهارم سوال اول

محتوای برخی از رجیسترها مانند Program Counter یا Stack Pointer توسط kernel handler قابل ذخیره‌سازی نیستند. چون خود آن‌ها هم نرم‌افزار هستند و تا CPU بخواهد آن‌ها را وارد مرحله اجرا کند، محتوای Program Counter و Stack Pointer عوض می‌شود. پس سخت افزار قبل از فراخوانی kernel handler، به طور خودکار محتوای رجیسترهای برخی از پروسس‌های متوقف شد را با push کردن آن‌ها در interrup stack حفظ می‌کند.

۵.۱ عنوان بخش پنجم سوال اول

پاسخ بخش دوم سوال در این قسمت نوشته می شود.

۶.۱ عنوان بخش ششم سوال اول

پاسخ بخش دوم سوال در این قسمت نوشته می شود.

۷.۱ عنوان بخش هفتم سوال اول

پاسخ بخش دوم سوال در این قسمت نوشته می شود.

۲ عنوان سوال دوم

سیستم کال ()fork یک پروسس جدید به نام پروسس child می‌سازد. پس از ایجاد پروسس child، هر دو پروسس(child و parentش)دستوراتی که پس از ()fork متناظرشان آمده اند را یک به یک اجرا می‌کنند. به این ترتیب پس از اولین ()fork، در مجموع ۲ پروسس، پس از دومین ()fork، در مجموع ۴ پروسس، پس از سومین ()fork، در مجموع هشت پروسس و ...، خواهیم داشت. درنتیجه پس از اجرای حلقه، تعداد childهای ایجاد شده برابر است با:

$$2 + 4 + \ldots + 2^{\log_2 n} = 2^{(\log_2 n)+1} = 2n$$

در مجموع با احتساب پروسس parent اصلی، ۲n پروسس خواهیم داشت.

۳ عنوان سوال سوم

در این قسمت با نحوه درج روابط و فرمول ها آشنا می شوید:

$$E = mc^2$$

۴ عنوان سوال چهارم

در این قسمت با نحوه درج اشکال آشنا می شوید:

شکل ۱: شکل شماره ۱

۵ عنوان سوال پنجم

۱.۵

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

int main()
{
```

```
        int n = 0;
        printf("Enter n: ");
        scanf("%d", &n);
        int stat;
        while(n != 1)
        {
                int rc = fork();
                if(rc < 0)
                {
                }
                else if(rc == 0)
                {
                        int m = n;
                        if(m % 2 == 0)
                        {
                                m /= 2;
                        }
                        else
                        {
                                m = 3 * m + 1;
                        }
                        return m;
                }
                else
                {
                        wait(&stat);
                        printf("%d, ", n);
                        fflush(stdout);
                        n = WEXITSTATUS(stat);
                        if(n == 1) printf("%d", n);
                }
        }
        return 0;
}
```

۲.۵

زیرا در UNIX/POSIX، exit code یک برنامه از نوع unsigned 8-bit تعریف شده است. به طور دقیق‌تر systemهای خانواده wait در UNIX نتیجه یک پروسس را به یک 32-bit integer کدگذاری می‌کنند. از این ۳۲ بیت برای اطلاعاتی همچون وقوع یا عدم وقوع dump core در پروسس، exit به دلیل یک سیگنال(و چه سیگنالی)، و ... تقسیم می‌شود. از این رو تنها ۸ بیت از ۳۲ بیت برای exit code باقی می‌ماند.

۳.۵

۶ عنوان سوال ششم

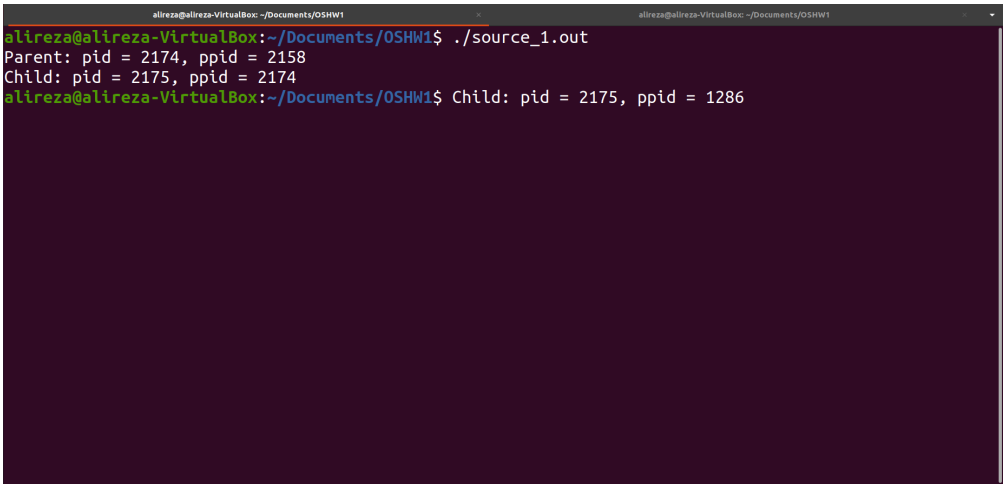
۱.۶

orphan process: پروسسی که parentش وجود ندارد(یا پایان یافته یا بدون اینکه برای متوقف شدن childش صبر کرده باشد، متوقف شده باشد)، orphan process نامیده می‌شود.

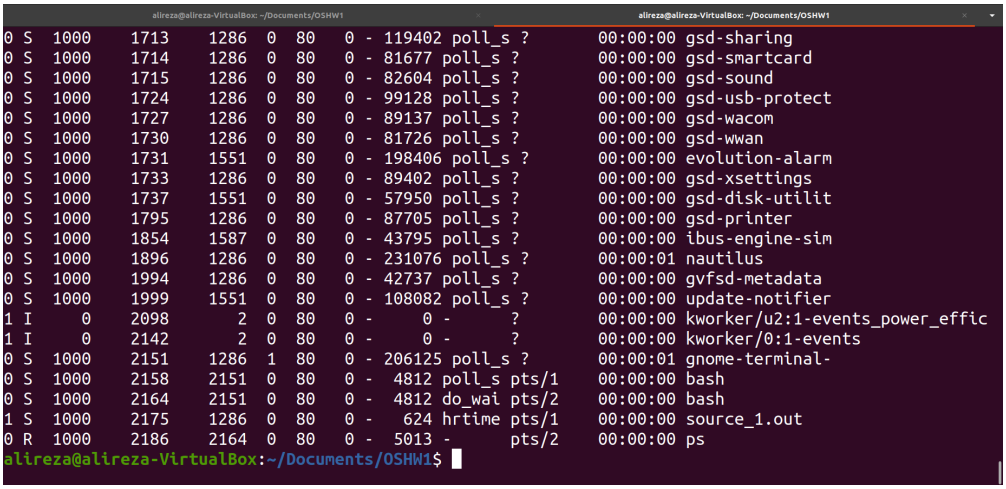
zombie process: پروسسی که اجرای آن پایان یافته است اما هنوز در جدول پروسس‌ها مقداری دارد که به پروسس parent نسبت داده می‌شود، zombie process نامیده می‌شود. یک پروسس child همواره پیش از پاک شدن از جدول پروسس‌ها به یک zombie process تبدیل می‌شود. پروسس parent، exit status پروسس child را می‌خواند و پروسس child از جدول پروسس‌ها حذف می‌شود.

۲.۶

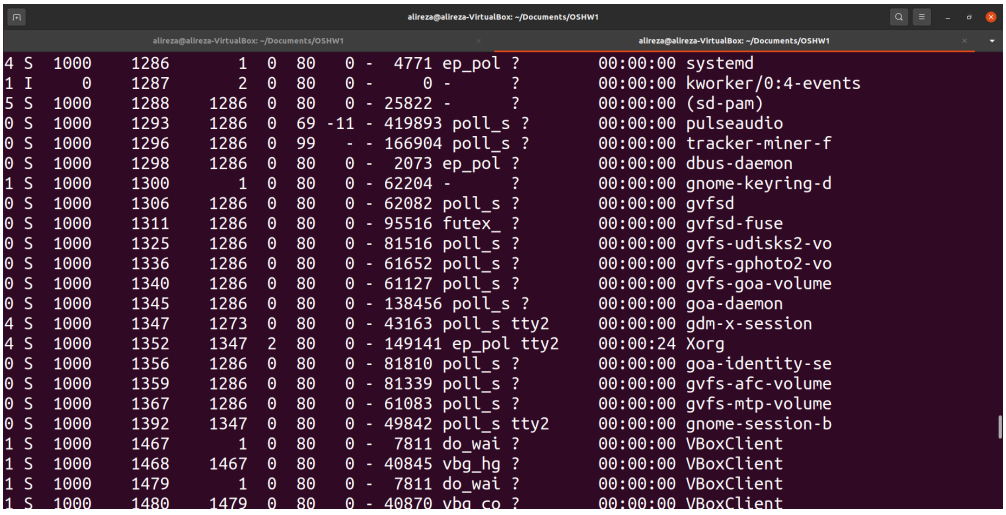
برنامه‌ی ۱ یک orphan process ایجاد می‌کند. چون پروسس child حدودا ۱۰۱ ثانیه پس از پایان یافتن پروسس parent به پایان می‌رسد. همینطور که در تصویر اول مشهود است، مقدار ppid پروسس child پس از ثانیه اول برابر ۲۱۷۴(pid پروسس parent) است اما پس از ثانیه سوم(پایان یافتن پروسس parent) این مقدار برابر ۱۲۸۶(pid پروسس systemd که نقش subreaper را برای پروسس orphan برعهده دارد و در قسمت بالای جدول پروسس‌ها قرار دارد) می‌باشد.



شکل ۲: اجرای برنامه‌ی ۱

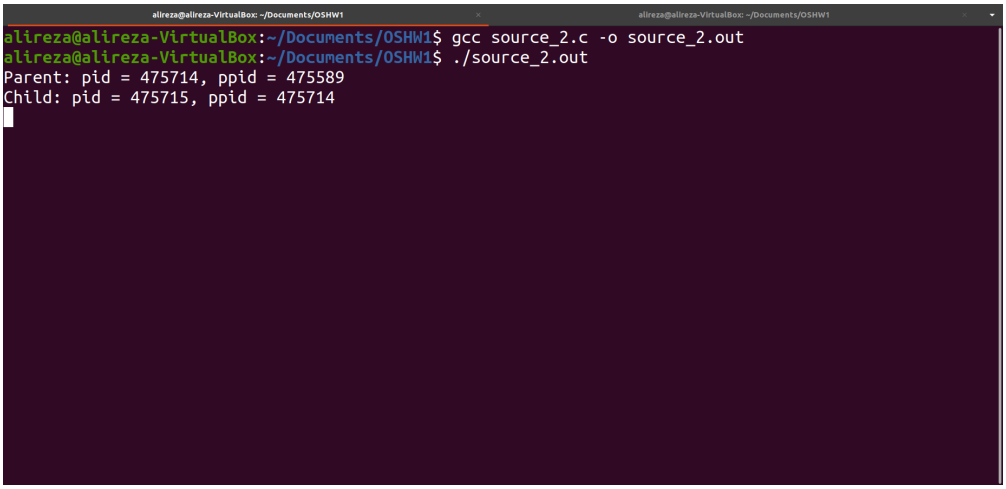


شکل ۳: جدول پروسس‌ها(قسمت پایین)

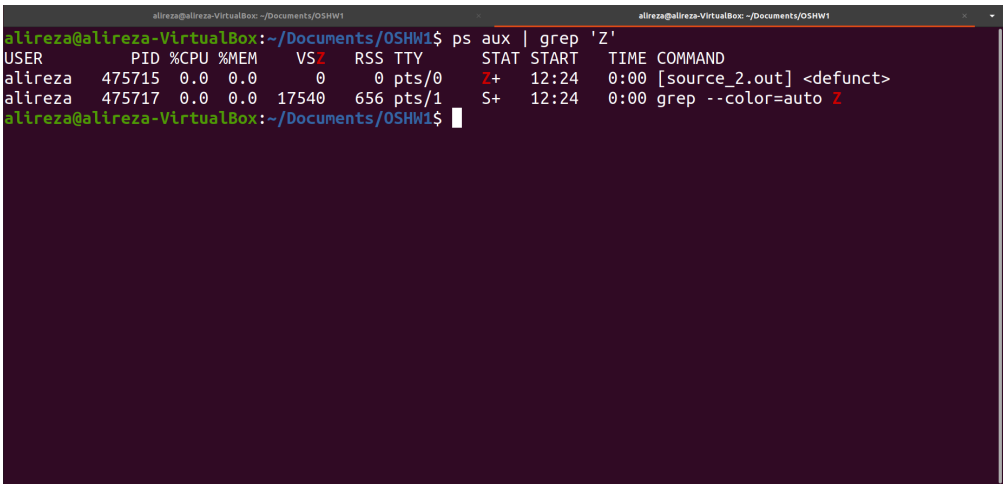


شکل ۴: جدول پروسس‌ها(قسمت بالا)

برنامه‌ی ۲ یک zombie process ایجاد می‌کند. چون پروسس parent حدودا ۹۹ ثانیه پس از پایان یافتن پروسس child به پایان می‌رسد. اگر در این بازه زمانی جدول پروسس‌ها را بررسی کنیم می‌بینیم که پروسسی با STAT Z وجود دارد که pidش برابر pid پروسس child است و این نشان‌دهنده این است که برنامه‌ی ۲ یک zombie process تولید کرده است. به تصاویر زیر توجه کنید:



شکل ۵: اجرای برنامه‌ی ۲



شکل ۶: جدول پروسس‌ها

تتتتتتتتتتتتتتتتتتتتتتتتتتت

۷ عنوان سوال جدول

در این قسمت با نحوه درج جداول آشنا می شوید:

خانه شماره ۱	خانه شماره ۲	خانه شماره ۳
خانه شماره ۴	خانه شماره ۵	خانه شماره ۶
خانه شماره ۷	خانه شماره ۸	خانه شماره ۹

جدول ۱: جدول شماره ۱

۸ عنوان سوال ششم

در این قسمت با نحوه درج انواع لیست ها آشنا می شوید:

۱.۸ عنوان بخش اول سوال ششم

- مورد اول
- مورد دوم

۲.۸ عنوان بخش دوم سوال ششم

۱. مورد شماره ۱

۲. مورد شماره ۲

۹ عنوان سوال هفتم

در این قسمت با نحوه درج برنامه ها آشنا می شوید:

```
#include <stdio.h>
int main()
{
        printf("Hello world!\n");
        return 0;
}
```

۱۰ عنوان سوال هشتم

در این قسمت با نحوه ارجاع به سایر منابع آشنا می شوید:
به صفحه درس سیستم عامل دکتر محمّدرضا حیدرپور ارجاع داده می شود [۱].

۱۱ ضمیمه

برای آشنایی بیشتر با LaTeX، با جست و جو در اینترنت منابع مفیدی خواهید یافت.

منابع

[1] http://mrheidar.ir/courses/operating_system.html