I.T.U.
Bilgisayar ve Bilişim Fakültesi
Bilgisayar Mühendisliği

Dersin Adı: İşletim Sistemleri

Dersin Kodu: BLG 312

Ad Soyad: Abdullah AYDEĞER

Numara: 040090533

Öğretim Görevlisi: Şima UYAR

Teslim Tarihi: 14.03.2012

1. Soruda verilen prog1.c adlı programı Xubuntu ortamında derleyip çalıştırdım. Derleme için prog1.c adlı dosyanın bulunduğu dizinde: “gcc prog1.c -o prog1” komutunu yazıp çalıştırma için de: “./prog1 yazdım. Bunlar yapıldıktan sonra alınan çıktı aşağıdaki gibidir:

Anne: Proses Kimlik Numaram: 12812
Anne: Cocugumun Proses Kimlik Numarasi: 12813
Cocuk: Proses Kimlik numaram: 12813
Cocuk: Annemin proses kimlik numarasi 12812
Anne: Sonlaniyorum...

Bu çıktılarla ilgili yorum yapmak gerekirse; Main programı başta anne olarak çalışmakta ve “fork()” çağrısı ile çocuk oluşturulmaktadır ve “fork()” çağrısından geri dönen değer “f” değişkenine hem anne hem de çocuk için atanmaktadır ki bu değer anne için yeni oluşturulan çocuğunun kimlik değeri, çocuk için ise “0” dır. Dolayısıyla f = -1 ise hatalı bir durum oluştuğu anlanacak veya f = 0 ise çocuğun else if bloğuna girecek ve oradaki işlemleri yapacak ve son(sıra anlamında değil de kod olarak son) olarak da anne için else bloğuna girecektir. Ekrana bastırılma sırası ile ilgili ise kesin olarak birşey söyleyememizle beraber birçok kez programı çalıştırmama ragmen yukarıdaki şekilde çıktı gözlemledim. Ekrana bastırılacak pid(proses kimlik değeri)’ lerden bahsetmek gerekirse bu değerler çekirdekteki proses tablosundan boş olan rastgele değerleri almaktadır ve program her çalıştırıldığında farklı bir değerler alınmıştır. Yukarıdaki örnek çıktıda anne için pid değeri 12812, “fork()” çağrısı ile oluşturduğu çocuğun pid değeri 12813 tür.

2. Anne proses için annesinin kimlik numarasını ekrana bastıracak şeklinde değiştirdiğimiz zaman prog1.c programını aşağıdaki satır çıktıya eklenmiştir:

Anne: Annemin Kimlik Numarasi: 4832

Bu satırın program her çağrıldığında aynı olduğu gözlemlenmiştir. Çünkü main programını oluşturan proses belli ve kimlik numarası sabittir. Programın farklı zamanda çağrılmalarına gore farklı değerler almamaktadır. Çünkü bu kimlik numarası aslında açılan terminalin kimlik numarasıdır ve sadece yeni bir terminal açıldığında değişebilir.

3. Programdan wait(NULL) satırı silindikten sonra tekrar derlenip çalıştırıldığında çocuk proses'in annesi öldürüldükten sonra devam ettiği, yani main programı bitmesine(ki bu da asıl programın bitmesi demek oluyor) rağmen çocuğun hala çalıştırılmasıdır. Çünkü wait(NULL) komutu anne prosesin yarattığı çocuk prosesi beklemesini sağlamaktadır. Bu bahsettiklerimi örnek bir çıktıyla göstermek gerekirse;

Anne: Proses Kimlik Numaram: 12849
Anne: Cocugumun Proses Kimlik Numarasi: 12850
Anne: Annemin Kimlik Numarasi: 4832
Anne: Sonlaniyorum...
abdullah@abdullah-VirtualBox:~/Desktop/İşletim Sis./1. Ödev/odev1_kod(1)$ Cocuk: Proses Kimlik numaram: 12850
Cocuk: Annemin proses kimlik numarasi 1

Main programı bittikten sonra da çocuk annesinin kendi kimlik numarasını doğru bastırmakta, ancak annesinin kimlik numarasını "1" olarak göstermekte ki bu değer gerçekteki değeri yansıtmamaktadır. Bunun nedeni ise anne prosesin ölmüş olmasıdır.

4. Verilen "prog1.c" programında int main satırından once bir global değişken tanımlandı ve ilk değeri sıfır olarak atandı. Daha sonra anne proseste bu global

değer ekrana basıldı ve bu global değer 2 yapılıp tekrar ekrana basıldı. Çocuk proses için ise yine global değer ekrana basılıp 3 yapılıp tekrar ekrana basıldı. Bu işlemlerin sonunda aşağıdaki çıktı gözlemlenmiştir.

Anne: Global ilk değer: 0
Anne: Global ikinci değer: 2
Çocuk: Global ilk değer: 0
Çocuk: Global ikinci değer: 3

Burada farkedilmesi gereken nokta global değişkenler de olsa bu değişkenler anne için kendi globali, çocuk için kendi globali mantığıyla işlemesidir. Bu yüzden global değer olmasına ragmen 2 proses farklı değerleri görmekte ve kullanmaktadır.

5. Main programından once bir global işaretçi tanımlayıp "fork()" çağrısından önce bu işaretçiye 25 değeri atadım ve adresiyle değerini ekrana yazdırdım. Daha sonra çocuk için bu global değişkenin ilk adres ve değerini ekrana bastırıp, global işaretçinin gösterdiği değeri 15 yapıp adresiyle birlikte tekrar ekrana yazdım ve benzer işi anne proses için de işaretçinin gösterdiği değeri 35 yapıp ekrana yazdırdım. Bu işlemler sonucunda ekranda gözlenen çıktı aşağıdaki gibidir:

Başlangıçtaki global değişken adresi: 0x92d0008 , değeri: 25
Anne: Başlangıçtaki global değişken adresi: 0x92d0008 , değeri: 25
Anne için: Global değişken adresi: 0x92d0008 , değeri: 35
Çocuk; Başlangıçtaki global değişken adresi: 0x92d0008 , değeri:25
Çocuk için: Global değişken adresi: 0x92d0008 , değeri: 15

Bu çıktıdan da görüldüğü üzere global bir işaretçi tanımlandığı zaman bu işaretçilerin adresi bunu kullanan prosesler için aynı olup gösterdikleri değer farklıdır. Bunun nedeni ise bu proseslerin kendi adres uzayları olmasıdır. Bu adres uzayı gerçek donanımdaki adres uzayı olmayıp prosesteki $0000100 adresi bizim donanımımızdaki $0001100 adresine denk gelebilmektedir. Bunu ise görüntü makina sayesinde kazanmaktayız ki görüntü makina işletim sisteminin bize sağlaması gereken temel görevlerinden biridir.

6. Soruda verilen p1.c programında bir ana prosesten 3 proses yaratılmakta ve her çocuk proses için “my_program” adlı alt program çağrısı yapılıp tmp global değişkeni 1 arttırılıp prosesin sırası ve tmp global değişkeni ekrana bastırılmaktadır. Diğer program olan p2.c’de ise 3 tane iplik yaratılmakta ve ipliklerin her biri yaratılırken “my_function” adlı alt program kullanılmakta ve tmp adlı global değişkenin değeri 1 arttırılıp ekrana yazdırılmaktadır. Her iki programa ait örnek çıktılar aşağıda verilmiştir:

P1.c

2: Value= 1
Main: Created 3 procs.
abdullah@abdullah-VirtualBox:~/Desktop/İşletim Sis./1. Ödev/odev1_kod(1)$ 1: Value= 1
0: Value= 1

P2.c

main(): Created 3 threads.
2: Value= 1
1: Value= 2
0: Value= 3

Bu çıktıları açıklarsak; P1.c deki prosesler bir adet global değişken üzerinde işlem yapmakta fakat bu global değişkeni her biri kendi adres uzayına kopyalayıp o kopyalanan değişken üzerinde işlem yapmakta ve dolayısıyla birbirinin değişkenlerini etkilememektedirler. Ancak P2.c deki oluşturulan iplikler de aynı global değişken üzerinde işlem yapmakta ve birinin arttırdığı değişken üzerine diğerleri işlem yapmaktadır. Yani her biri aynı değişkeni(aynı bellek adresini) etkilemektedir çünkü ipliklerin kendi adres uzayı olmayıp yaratılmış oldukları prosesin adres uzayını kullanırlar. Bu yüzden beklenen çıktı gözlemlenebilmiştir.