MİNİTALK

Server dosyası main tarafı

mr. maksut tez bilgi:

16 bit = ö - ingilizce karakter dışındaki tanımlı olmayan karakterler

32 bit = 😔

8 bit = a

signals

sinyaller için komutlar

kill -STOP [PİD] ----> bu komut servere durdurur.

kill -CONT “” ---> devam ettirir ama benim kodda ettirmez.

ayrıca sinyaller kullanılırken printf kullanılması tavsiye edilmez çünkü çökme ihtimali çok yüksektir.

kill -l ---> tüm sinyalleri verir.

ide kullanrak sigusr sinyallerinin üzerine gelindiğinde 30 ve 31 olması kill -l komutunun kullanılırarak bakıldığında 30 ve 31 olarak gösrerilmesinin sebebidir.

```
k2m18s09:Desktop bsoykan$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGEMT       8) SIGFPE
 9) SIGKILL     10) SIGBUS      11) SIGSEGV     12) SIGSYS
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGURG
17) SIGSTOP     18) SIGTSTP     19) SIGCONT     20) SIGCHLD
21) SIGTTIN     22) SIGTTOU     23) SIGIO       24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGINFO     30) SIGUSR1     31) SIGUSR2
k2m18s09:Desktop bsoykan$
```

```c
int   main(void)
{
    ft_putstr_fd("PID: ", 1);
    ft_putnbr_fd(getpid(), 1);
    ft_putstr_fd("\n", 1);
    signal(SIGUSR1, handler); // her bir bit için program dinleme yapar her bir bit için çalışır ve handler fonksiyonun
    signal(SIGUSR2, handler); // her bir bit için program dinleme yapar her bir bit için çalışır ve handler fonksiyonun
    while (1)
        pause();
    return (0);
}
```

signal kütüphanesi kullanılarak getpid yazdırılır.

- **GETPİD: process İD diğer her işlem için PİD vardır. işlemlerin birbiriyle karışmaması için benzersiz bir işlem kimliği yaratılır.**

signal fonksiyonu kullanılarak SIGUSR1 ve 2 sürekli dinlenir. Bu belirlenmiş PİD ye 1 veya 0 sinyali gelmesi demek ikisinden birisi geldiğinde handler fonksiyonuna gönderir.

-while(1) ise programın sürekli açık kalıp sinyal dinlemesini sağlamak.

server dosyasındaki handler fonksiyonu işlevi

```c
void  handler(int signum)
{
    static int  letter;
    static int  power;

    if (!power)
        power = 128;
    if (signum == SIGUSR1) // eğer gelen sinyal SIGUSR1 ise 1 ile kaydırma yapar ve toplar.
        letter += power; // 1 ile kaydırma yapar ve toplar.
    power /= 2; // power = 64, 32, 16, 8, 4, 2, 1 şeklinde çıkartma yapar.
    if (power == 0){
        ft_putchar_fd(letter, 1);
        power = 128;
        letter = 0;
    }
}
```

static kullanılmasının sebebi her bit için client dosyasından gelen sinyallerin static olan değişkenlerdeki veriyi kaybetmek istememesi.

bu fonksiyona sadece 0 ve 1 değeri gönderilir, signum değeri gönderilen sinyali temsil eder eğer 2. sinyal gönderildiyse değer 0'dır. bu durumda o biti geçer ve default olarak 0 değerinde kalır power 2'ye bölünür ve bu bölüm sonucunda bir bit sağa geçilmiş olunur. Eğer 1 sinyali gönderilse o sıradaki biti power ile toplanır ve örnek verilecek olursa şöyle bir sonuç açığa çıkar:

diyelim ki A harfinin sinyalleri geliyor:

- **ilk önce power değerlerinin yani 128 ve ikiye bölünen katların bit değerleri:**

```
power = 128 = 0b10000000
power = 64 = 0b01000000
power = 32 = 0b00100000
power = 16 = 0b00010000
```

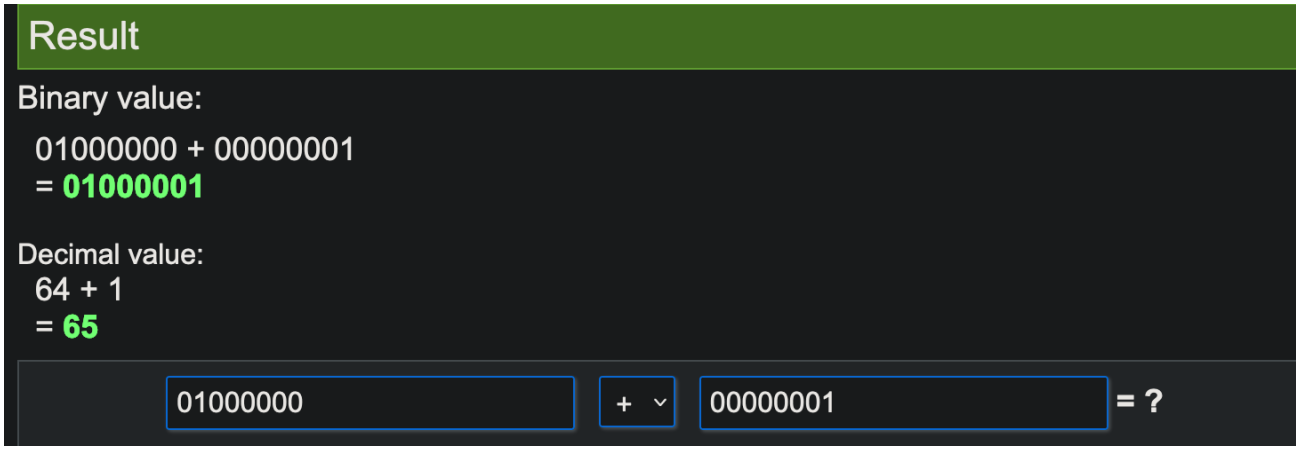
```
1   power = 8 = 0b00001000
2   power = 4 = 0b00000100
3   power = 2 = 0b00000010
4   power = 1 = 0b00000001
```

A = 65 = 0b01000001 ilk gönderilen sinyal en soldaki değer olacak 0 olduğu için power 2'ye bölünerek devam edecek. 1. aşamada letter değişkeni 8 adet 0 değerinden oluşmaktadır.

2.aşamada A değerinin soldan 2. değeri 1 olduğu için handler fonksiyonun içine 1 değeri olarak girecek, power değeri 64 olduğu için tam olarak A değeriyle aynı yere yani soldan ikinci basamağına 1 değerini koyup gerisini de sıfıra eşitleyecek. Letter değişkeni bu durumda 64 değerinin bit değerinin aynısı olacak.

3.aşamada da aynı şekilde 8. aşamaya kadar 0 olarak devam edecek handler fonksiyonu çalıştıralacak böylece fonksiyonda sadece power değişkeni bölünmüş olacak. son değer 1 olacak.

8.aşamada handler fonksiyonu A değerindeki gibi 1 değeri gönderince signum değeri de 1 olduğundan dolayı 8. aşamaya gelmiş olan letter değeri en son 2. aşamada 64 değeriyle toplanıp 64 değerinin bit değerini almıştı. 64 değerini bit sayısı 1 değerinin bit sayısıyla toplanıyor çünkü 1 sinyali gönderilmiş. bu duruma örnek olarak:



- **64 + 1 = 65, ASCİİ =) 'A' = 65;**

Power değişkeni bu noktada tekrar bölünerek 0 değerini alır ve if koşuluyla kontrol edilir. sebebi ise biz 1 byte yazdırmak istiyoruz ve 1 byte 8 bit

değerindedir. Bu sebepten dolayı if koşulu ile 1 bytelık okuma bitmiş mi kontrol edilir. Eğer 1 byte yani 8 bit bittiyse letter değişkeni yani yazdırılmak istenilen bit düzeni çıktıya gönderilir. Yazdırıldıktan sonra statik olan değşikenler tekrardan default ayarlarına eşitlenir.

Bu noktaya kadar gelinen kısım server dosyasıydı şimdi bu bitlerin nasıl gönderildiğini açıklayalım:

CLİENT DOSYASI SADECE MAİN FONKSİYONUNDAN OLUŞMAKTADIR, SIRASIYLA:

```c
int main(int ac,char **av){

    int i;
    int pid;
    int power;
    char *str;

    i = 0;
    power = 128;
    if (ac < 3){
        ft_putstr_fd("ErRor: arguments not enough!!!\n", 2);
        return (-1);
    }
    else if (ac > 3){
        ft_putstr_fd("ErRor: arguments too much!!!\n", 2);
        return (-1);
    }
```

server dosyasındaki gibi power değişkenimiz kullanılır ikisinde de soldan sağa doğru basamak basamak kontrol etmemizi sağlar.

str değişkeni iste argümandan gelecek değerleri temsil eden bir pointer değişken.

pid değişkeni argümandan alınan string değerinn int değişkene atanması için.

i değişkeni ise str değişkenini içindeki indexte gezmek için.

argüman kontrolleri yapılır argümanın istenilen sayıda olması istenilir.

```c
    pid = ft_atoi(av[1]);
    str = av[2];
    while (str[i]){
        while (power >= 1){
            if (str[i] & power) //if içerisindeki koda göre gönderim yapar
                kill(pid, SIGUSR1);
            else
                kill(pid, SIGUSR2);
            power /= 2;
        usleep(100);
        }
        power = 128;
        i++;
    }
    return (0);
}
```

döngü içinde döngü açılarak str değişkenininnn sonuna kadar devam eder. fakat her bir byte değerinde 8 bit olduğu için iç döngüde 8 kere devam edip i değerini temsil ettiği byte değerini bitirmiş olur. sonraki byte değerinde de aynı işlemi yapmak için power değişkenini yine 128 değerine ayarlar.

i değerinin temsil ettiği bit düzeni & operatörüyle karşılaştırılır. örneğin :

```
A = 65 = 0b01000001 & 128 = 0b10000000;
```

Bu karşılaştırma sonucunda if koşulu 0 olacağı için else değerini kill fonksiyonuyla gönderir.

else olarak gönderildiği için SİGUSR2 0 olarak server dosyasına pid üzerinden iletilir.

sonraki biti kontrol etmek için power 2' ye bölünür usleep kodu ise 100 mikrosaniye delay koyar bunun sebebi linux sinyallerinde gönderilen sinyallerin sıraya konulmadan hızlı hızlı ulaşması, bir sinyal önceki gönderilenden daha hızlı giderse bu durumda sinyal karışmaları ıkabilir bunun önüne geçmek için usleep kullanılur.

sonrali aşamada:

```
A = 65 = 0b01000001 & 64 = 0b01000000;
```

Bu aşamada ise if 1 olacağından gönderilen sinyaller de 1 olarak gönderilir.

Özet:

argüman üzerinden alınan pid ve string değeri farklı değerlere eşitlenir ve bu değerler üzerinden işlem devam eder. ilk olarak eksik veya fazla argüman girip programın düzgün çalışmamasını önlemek amacıyla argüman sayısı kontrol edilir. pid argümandan girildiği için string olarak yer tutacaktır bunu atoi kullanarak sayıya dönüşüm sağlanır. sonrasında gönderilmek istenen str değişkenim için her bir indexsine yani byte değerine ulaşmak için index üzerinde işlem yapılır. Her bir index 1 byte olduğu için 8 bit boyutundadır ve bu her 8 bit üzerinde sayının değeri 0 ve 1 rakamlarıyla gösterilmektedir. Bu rakamları bit bit olarak göndererek yazdırma işlemi yapılır. Her bir bit client dosyasında karşılaştırma operatörü kullanılınarak karşılaştırlır ve karşılaştırmaya uyan değerler SİGUSR1 ve SİGUSR2 sinyalleri olarak server dosyasına gönderilir, gönderilen bu sinyaller 1 ve 0 rakamlarını temsil ederler bu döngü 8 kere sağlanacağı için her seferinde server tarafına bağlı olmayan ayrı bir power değişkeni üzerinde static olarak işlem yapılır, statik olarak yapılmasının sebebi ise 8 bit boyunca her seferinde client dosyasından sinyal gönderilmesi ve farklı dosyalarda işlem yaparken değişkenin kaybolmasını önlemektir. her seferinde client dosyasından 0 ve 1 değişkeni gönderilir bu aşama 8 bit boyunca devam eder, bunun bitip bitmediğinin anlaşması iiçin server tarafında power değişkeninin 0 olup olmadığı kontrol edilir eğer değişken 0 ise client tarafındaki 8 bitlik değişken bitmiş demektir. server tarafına gelen sonraki bitleri kontrol etmesi için değerler default olarak yeniden belirlenir ve döngü böylece devam etmiş olur. server tarafında client tarafından gelen sinyalleri dinlemeye yarayan değişken ise signal fonksiyonlarıdır. gelen sinyalleri handle fonksiyonun gönderirler ve sonsuz while döngüsü de oludğu için program sonlanana kadar PİD kanalından gelen belirtilen sinyalleri dinlemeye devam edeler.

makefile için

```
server: server.o libft
	$(CC) -o $@ $< -Llibft -lft
```

$(CC) ---->> derleyici başına konur ve derleyici olduğunu belli eder.

-o $@ ----->> derlenen her hedef dosyanın adının dosyanın kendi adı olduğunu belirtir.

$< -----> derlenen her kaynak dosyanın adını belirtir.

--Llibft -lft ----> libft adlı kütüphaneyi -lft komutuyla linkleyerek derlenecek her bir dosya için kullanmaya yardım eder.

% → herhangi bir dosya adını kontrol eder.

$? ---> kullanılacak kaynak dosyaların isimlerini belirler = .0 ve .c olanlar.

soru ve cevaplar 40 soru:

İlk kod parçası ne iş yapar? İlgili sinyalleri (SIGUSR1 ve SIGUSR2) nasıl ele alır?

bu sinyaller kullanıcı sinyalleridir, kullanıcı sinyallerini kullanmak için string değerinin 1 byte değerini alır ve onu 8 bit olarak işler. her bit içn 0 ve 1 değerini server dosyasına gönderir.

handler işlevinin ne iş yaptığını açıklayın. İlgili sinyalleri nasıl işliyor ve neden usleep(100) kullanılıyor?

handler fonksiyonu gelen sinyallerin türüne göre yani 0 veya 1 olup olmadığına göre çalışır ve aldığı bu değerleri static değişken olan değişkenlere atar. atadığı bu değişkenlerde 8 bit sırası boyunca alınan bitleri birleştirir. doğru bit olup olmadığını ise client dosyasındaki power değişkeniyle aynı sırada giderek yapar. letter değişkenine aynı bit sırasını işler (toplayarak, bkz: bit toplama).

main işlevi neler yapar? signal işlevi ne işe yarar ve hangi sinyalleri ele alır?

main tarafında sırasıyla, sistemdeki boş olan pid alınır ve bu pid ekrana yazdırılır, ardından signal fonksiyonu kullanılır bu fonksiyon istenilen sinyali sonsuz döngü

içindeki pause yardımıyla sürekli dinler. istenilen sinyal geldiğinde ise bu sinyali gönderilmek istenilen fonskiyona gönderilir.

İkinci kod parçası ne iş yapar? Hangi komut satırı argümanlarına ihtiyaç duyar ve bu argümanlar neyi temsil eder?

client dosyasındaki argümanlar pid ve string olması gerekir, pid server dosyasından alınan yeri temsil eder ve bu işlemci sırası numarası ikisinde de doğru olmak zorundadır. argümanlardan string olarak geldiği için sayıya dönüştürülür. sonrali argüman ise alınan değeri string olarak alır ve bitleri kullanarak server dosyasına gönderir.

checker işlevi ne iş yapar ve hangi durumlarda hangi hataları kontrol eder?

checker değeri argüman sayılarını kontrol eder ve iki farklı hata mesajı yayınlar, bunlar arügmanın fazla veya az olduğunu belirtir. Bir diğer kontrol ise pid numarasının -1 aldığı durumlarda çökmesini engellemek için kullanılır.

İkinci kod parçası, belirli bir işlem kimliği (PID) ve bir metin dizesi alarak ne iş yapar?

aldığı bu argümanları döngü içerisinde bitwise operatörü kullanarak bit boyutunda karşılaştırır ve karşılaştırdığı değerleri true veya false olarak işler. işledikten sonra ise 30 veya 31. sinyalleri kullanrak duruma göre sinyalleri gönderir.

İkinci kod parçasındaki kill işleminin amacı nedir? SIGUSR1 ve SIGUSR2 sinyalleri neden kullanılır?

bu sinyaller unix sistemin içinde gelir ve bu sinyalleri kullanıcı sinyali olarak kullanabiliriz. diğer 29 sinyalin ise her birinin ayrı bir görevi ve işlevi vardır. fakat 30 ve 31. sinyal ise kullanıcılara özel bırakılmıştır bu sinyalleri her türlü işlev için kullanabiliriz.

İkinci kod parçasında neden usleep(100) kullanılır?

bu 100 mikrosaniye bekleme yapar, bunun amacı işlemlerin farklı hızlarda işlenip üst üste binmesini ve dolayısıyla istenilenden farklı çıktılar almayı engeller.

İlk kod parçası ve ikinci kod parçası arasındaki iletişim nasıl gerçekleşir?

ilk kod parçası kill fonskiyonu kullanarak sinyali gönderir ve gönderilen bu sinyali signal fonksiyonuyla diğer dosyadan dinlediğimiz için gönderilen dosya işlenir.

Bu kodlar üzerine eklemeler veya iyileştirmeler yapabilir misiniz? Önerilen geliştirmeler neler olabilir?

kill fonksiyonunun üzerinde gönderilip gönderilmediğini anlamak için -1 kontrolü yapılabilir, eğer sinyal göndermesinde bir problem çıkarsa hata mesajı yayınlanabilir.

İlk ve ikinci kod parçaları arasında hangi tür hatalar veya sorunlar oluşabilir?

çok yüksek veri aktarımlarında fazla iş biriktiği için biriken fazla işi handler edemeyebilir, bu durumda da çıktıda bozuntular meydana gelir.

Bu kodların çalışma mantığını baştan sona açıklayın.

clientten sinyal alır serverda işler.

İkinci kod parçasındaki power değişkeni neyi temsil eder ve neden 128 ile başlar? Bu değişken ne işe yarar?

power değişkeni 128 değerini almıştır çünkü 128 değeri binary olarak 1000000 değerini temsil eder ve her seferinde power /= 2; yapılarak yarısına indirgilenir.power değişkeninin yarısına indikten sonra 64 değerinin binart değeri 01000000 olur ve böylece devam eder. istenilen byte sabit kalır ve byte değerinin binary şekli bitwise operatörüyle her seferinde kontrol edilir. Eğer ki 1 olan yerler true döndürüyorsa alınan sinyal 1 olarak kill yardımıyla gönderilir. bu 8 aşamada devam eder.

İlk kod parçasındaki ft_putchar_fd işlevi ve ikinci kod parçasındaki ft_putstr_fd işlevi ne iş yapar? Bu işlevler özel bir kütüphane veya modülde nasıl tanımlanır?

putchar fonksiyonunda alınan değer bir byte olmak zorundadır ve byte byte yaazırılan işlemlerde genellikle kullanılır. pustr ise string bastırmak için kullanılır ve daha kolay bir kullanım sağlar. fd işlevi ise standarrt dosya yönlendirmesinde kullanılan işlemdir.

İkinci kod parçasındaki str değişkeni ve döngü, hangi amaçla kullanılır? Bir metin dizesini bitiş karakterine kadar nasıl işler?

her bir byte için 8 döngü boyunca devam eder. her bir byte değerinin 8 bit değeri gönderildikten sonra sonraki byte değeri varsa sonrakine geçip olan byte değerini yine aynı şekilde işler ve 8 bitin her birini döngü içinde sürekli gönderir.

İkinci kod parçasında, hangi durumda checker işlevi 0 döndürür ve neden?

argüman durumlarında sıfır döndürür ve if değerinde 1 olarak yani bir problem olarak algılanır alınan değer de eror log olarak işlenir.

İlk kod parçasında, işlemi sonsuz bir döngü içinde tutma nedeni nedir? Hangi şartlar altında bu döngüden çıkılır?

server dosyasındaki sonsuz döngünün sebebi isteyerek kapatılana kadar sürekli sinyal dinlemesini istememizdir.eğer döngü olmazsa program sinyali alır almaz sonlanır. pause tek başına sonsuz şekilde sinyali dinlememiz için yeterli değildir.

İlk kod parçasında, SIGUSR1 ve SIGUSR2 sinyalleri ne tür olayları temsil edebilir? Hangi senaryolar için bu sinyaller kullanılır?

bu sinyaller kullannıcının custom sinyal oluşturması için ayrılmıştır ve her isteğe göre handler edilebilir.

Bu kodlar, bir iletişim protokolünün bir parçası olarak mı kullanılıyor? Eğer öyleyse, bu iletişim protokolü nedir ve nasıl çalışır?

iletişim protokolleri pid üzerinden gerçekleşir istenlen pid numarası kullanılarak istenilen uygulama üzerinde istenilen sinyallerle işlem yapılabilir.

İkinci kod parçasında, kill fonksiyonunun hangi işlemi hedeflediğini nasıl belirliyoruz? pid değişkeni bu işlemi nasıl tanımlar?

kill fonksiyonu istenilen sinyali istenlen pid numarasına gönderiri.

İkinci kod parçasında, işlemi hedeflemek için kill işlevini nasıl kullanıyoruz? SIGUSR1 ve SIGUSR2 sinyalleri nasıl kullanılıyor?

bu sinyaller kodda sinyaller 1 ve 0 olarak if koşuluna bağlı olarak gerçekleşir.kill fonksiyonuyla sinyal pid ile programlar arası aktarılır.

İlk kod parçasındaki handler işlevinde power ve letter değişkenleri neden static olarak tanımlanmıştır? Bu nedenle static kullanmanın avantajları nelerdir?

static olarak tanımlanmasının needeni hafızada kalıcı bir yer alması ve programın işlem görmediği halde default ayarlarına dönmesini engeller ve kaldığı yerden devam etmesine yarar.eğer static kullanılmazsa kod çalışmaz

İkinci kod parçasında, ac (argüman sayısı) ve pi (PID) değerlerinin kontrolü için checker işlevi kullanılıyor. ac ve pi değerleri nasıl belirlenir?

ac değeri girilen argüman değerini programdan alarak tutan değişkendir, pid ise argüman olarak alınır ve int bir değeree ft_atoi kullanılarak dönüştürülür.

İlk ve ikinci kod parçaları arasında iletişim nasıl sağlanır? İlk kod parçası sinyal alırken, ikinci kod parçası sinyal gönderir. Bu süreç nasıl gerçekleşir?

bu süreç yardımcı fonksiyonlar kullanrak gerçekleşir, fonksiyonları kullanırken biri gönderir biri dinler yakalar.,

İlk kod parçasındaki pause işlevi ne işe yarar ve neden sonsuz bir döngü içinde kullanılır?

bu yardımcı fonksiyonun sonsuz döngü içinde kullanılmasının sebebi sürekli sinyal dinlemek istemesi eğer bu while içinde olursa sinyal her zaman dinlenir.while içinde yazılmazsa kod sadece 1 kere sinyal dinler.

İkinci kod parçasındaki döngü, her karakteri nasıl işler ve ilgili sinyalleri gönderir? İşaretlerin gönderilme sırası ve zamanlaması nasıl çalışır?

client programında kod içinde string değerinin her bir byte değeri içn işleme girer ve döngü içindeki döngüde her bir bit için sinyal değeri gönderir. bu bitler server tarafında işlenerek bit haline getirilir.

İlk ve ikinci kod parçaları, hangi senaryolarda kullanışlı olabilir? Bu kodlar, hangi tür uygulamalar için uygundur?

bu tür kodlar genel olarak bir programa ne yapması gerektiğini belirtmek için kullanılabilir tamamen hayal gücüne kalmış bir kod parçası.

Kodun güvenlik açısından potansiyel sorunları olabilir mi? Örneğin, kullanıcıların

hatalı veya kötü niyetli girdilerle sistemi etkileyebilme olasılığı var mıdır?

kodun manüpile edilerek bozulması sonucu tüm pc yi kitleyebilecek bir pid bugu oluşabilir.

İlk ve ikinci kod parçalarında kullanılan sinyallerin SIGUSR1 ve SIGUSR2 olması bir zorunluluk mu, yoksa başka sinyaller de kullanılabilir mi? Hangi durumlar için farklı sinyaller kullanılabilir?

bu sinyaller sadece kullanıcının kullanması için ayrılmıştır istenilirse farklı sinyaller de kullanılabilir fakat bu sefer program beklenmedik davranışlar sergiler.

İkinci kod parçası, veri iletimi için her karakter arasında herhangi bir senkronizasyon işlemi yapar mı? Verinin doğru bir şekilde iletildiğinden nasıl emin olunur?

verinin doğru bir şekilde emin olunması iiçn her bir bit boyutu boyunca yazdırma işlemi gerçekleştirilir.

İlk ve ikinci kod parçaları, hangi işletim sistemlerinde çalışabilir? Sistem bağımlılıkları nelerdir?

kütüphane ve program dili olan her işletim sisteminde çalışabilir.

İkinci kod parçasındaki argümanları doğru bir şekilde kontrol etmeyen veya kullanmayan bir kullanıcı, bu kodları nasıl kötüye kullanabilir veya hatalara yol açabilir?

doğru bir şekilde kontrol edilmezse eğer pid numarası üzerinden ya da amaca göre mesaj üzerinden kötüye kullanılabilir.

İlk kod parçasında kullanılan usleep fonksiyonu, iletişim hızını sınırlamak için mi yoksa başka bir amaçla mı kullanılıyor?

usleep kullanılmasının sebebi çıktıların doğru bir şekilde çıktı vermesidir. iletişim hızını sınırladığı gibi kontrollü bir çıktı almayı sağlar.

İlk kod parçasındaki sinyal işleme yöntemi ve ikinci kod parçasındaki sinyal gönderme yöntemi, alternatif iletişim yöntemleriyle (örneğin, soketler) nasıl karşılaştırılabilir?

iki karşılaşştırmada da benzer durumlar geçerlidir çünkü ikisi de sinyallerle çalışır ve bu sinyaller yardımıyla veri aktarımı gerçekleşir. birinde fiziksel diğerinde sanal şekilde olur.

İlk kod parçası, hangi tür uygulamalarda sinyal işleme için kullanılabilir? Hangi senaryolarda bu tür bir iletişim mekanizması tercih edilir?

programlar arası işlemlerde hızlı ve etkili bir yol olarak bu iletişim yolu tercih edilebilir, otomatikleştirilebilir.

İlk kod parçasındaki handler işlevinin sinyal işleme için kullanıldığı bu örnekte, hangi tür verilerin iletilmesi için kullanılabilir? Bu tür bir veri iletiminin avantajları ve dezavantajları nelerdir?

bu tür veri iletiminde dosya aktarımı çok mantıklı olamaz çünkü güvenlik açığı oluşturabilecek çok fazla açık barındırır. fakat basit metinsel mesajlar veya bir programa gönderilecek basit ikazlar için kullanılabilir.