**Alıştırma**

1. Aşağıdaki C kod parçası ekrana ne yazar? Child process’ in PID=90 olarak alabilirsiniz.

int main() {

pid\_t pid = fork();

printf("Merhaba: %d\n", pid);

return 0;

}

|  |
| --- |
| Merhaba:90  Merhaba:0  Veya  Merhaba:0  Merhaba:90  Veya  Merhaba:90  Parent veya child’dan hangisinin önce geldiğine bağlı. Parent önce biterse child yazamadan kapanır. |

**Sorular**

1. Aşağıdaki program ile kaç tane görev oluşturulur?

int main() {

for (int i = 0; i < 3; i++) {

pid\_t pid = fork();

}

return 0;

}

|  |
| --- |
| 8 tane oluşur. |

1. Aşağıdaki kod çalıştırılırsa ne olur? **(Kodu denemek zorunda değilsiniz)**

int main(void)

{

while(1) {

fork();

}

}

|  |
| --- |
| Sonsuz döngüde Parent’ın child’ları ve child’ların child’ları oluşmaya devam eder sanıyorum ki hafıza dolana kadar. |

1. Aşağıdaki C kodunun çıktısı nedir? Nedenleriyle açıklayınız.

int main() {

int sayi = 7;

pid\_t pid = fork();

printf("Sayi= %d\n", sayi);

if (pid == 0)

sayi = 6;

return 0;

}

|  |
| --- |
| Sayi= 7  Sayi= 7  Sadece child’ın olduğu Sayi değişkeni 6 olur ama önceden ekrana yazıldığı için ikisi de aynı olur.  Veya eğer Parent yeterince hızlıysa:  Sayi= 7  Yazılır ve program son bulur. |

1. Aşağıdaki C kodunun çıktısı nedir? Nedenleriyle açıklayınız.

int main() {

int\* sayi = malloc(sizeof(int)\*1);

\*sayi = 7;

pid\_t pid = fork();

printf("Sayi= %d\n", \*sayi);

if (pid == 0)

\*sayi = 6

return 0;

}

|  |
| --- |
| Bir önceki soruyla aynı çıktılar alınır burada tek fark kendi integer değişkenimiz olan sayi pointerına bellekte kendimiz yer ayırarak(4 bytes) oluşturuyor olmamız. |

1. Aşağıdaki duruma göre C kodunun çıktısı nedir?

void onSayiYaz(int \*arr)

{

int i;

printf("\n");

for(i=0;i<10;i++)

{

printf("%d",arr[i]);

}

exit(0);

}

int main()

{

int \*arr, i;

arr = (int \*) malloc (sizeof(int));

arr[0] = 0;

for(i=1;i<10;i++)

{

arr = (int \*) realloc( arr, (i+1) \* sizeof(int));

arr[i] = i;

if (i == 7)

{

pid\_t pid = fork();

if (pid == 0)

{

onSayiYaz(arr);

}

}

}

OnSayiYaz(arr);

return 0;

}

|  |
| --- |
| 0123456789  0123456700 |

1. Aşağıdaki koda göre muhendis.txt dosyasına ne yazılır?

int main()

{

int fd;

fd = open("muhendis.txt", O\_CREAT|O\_TRUNC|O\_WRONLY, 0666);

if(!fork()) {

write(fd, "SDÜ ", 6);

exit(0);

}

else {

int status;

wait(&status);

write(fd, "BilMüh\n", 6);

}

return 0;

}

|  |
| --- |
| SDÜ BilMüh  Child = 0(false) olduğu için değili 1(true) ve ilk if’in içine girer. Parent true olduğu için değili false olur ve else’in içine girer ve iki write() komutu da çalışır. |

1. Fork() işleminde child process’ in adres uzayı parentları ile tamamen aynıdır. Eğer siz bir işletim sistemi tasarlasaydınız, adres uzayı kopyalama işlemini daha etkin/ etkili yapmak için neler yapardınız.

|  |
| --- |
| Child her şeyi kopyalıyor belki child sadece kullanacağı komut satırlarını kopyalasaydı büyük boyuttaki dosyalar için daha avantajlı olabilirdi mesela eğer child’ın kullanacağı komut satırlarını biliyorsak komutu yazarken bir işaretçi gibi bir şey koyabiliriz printf(“%d”, sayi)\_; gibi böylece child’ların sadece bu komut satırlarını kullanacağını anlar ve sadece bu komut satırlarını kopyalar ve diğerlerine bakmakla uğraşmaz. |