Subject

Date. Month. Year.

۱) الف) فرآیند فرزند می تواند منابع خود را مستقیماً از سیستم عامل دریافت کند یا اینکه دسترسی آن به بخشی از منابع فرآیند پدر، محدود شده باشد

ب) ① فرآیند فرزند از برخی منابعی که در اختیار او قرار داده شده بود، بیش از حد استفاده کرده است ② وظیفه محول شده به فرآیند فرزند دیگر مورد نیاز نیست

③ فرآیند والد در حال اتمام است و سیستم عامل به فرزند اجازه نمی دهد در صورت از بین رفتن پدر به کارش ادامه دهد

ج) درست است زیرا pid که خروجی دستور fork است شامل ۳ حالت می باشد که اگر کوچکتر از صفر باشد یعنی خطایی در تولید فرزند بوجود آمده اگر صفر باشد فرآیند فرزند بوجود آمده و اگر بزرگتر باشد یعنی id فرآیند والد است.

دستور wait(NULL) در قسمت فرآیند والد برای این است که پدر تا اتمام کار فرآیند فرزندها صبر کند. و سپس آنها به خط exit(n) خروج کنند.

در نهایت به تعداد فرزندها در خروجی "child" و در نهایت "Parent" نوشته می شود

۲) $(8+1) \% size = 9$ ← شروع

Full ⇒ $(out + 1) \% size = in$ ← پایان

حداقل تعداد size برابر ۱۰ است ⇒ $9 \% size = 9$

تا این شرط برقرار باشد

حداکثر تعداد item ها $size - 1$ است بنابراین حداکثر ۹ item می توانیم قرار دهیم

SANA

3) حافظه shared memory محدوده ای از فضای آدرسی وجود دارد که فرآیند تولید کننده آن را ساخته است. فرآیند دیگر که می خواهد از این shared memory استفاده کند باید این فضای آدرسی را به آدرس خودش بیندد. در حالت shared memory محدودیتی که OS برای جلوگیری از دسترسی یک فرآیند به حافظه فرآیند دیگر انجام می داد را برطرف کرده است

در واقع در این روش هر دو فرآیند می توانند در حافظه مشترک بخوانند یا بنویسند و دسترسی داده ها و کانال آنها بر عهده OS نیست و خود فرآیندها باید آن را مدیریت کنند

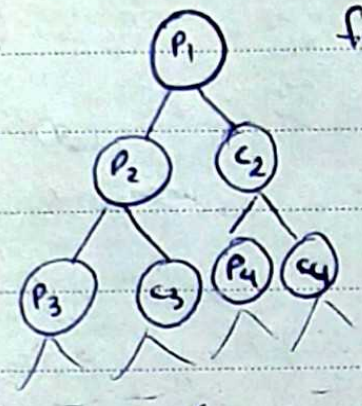
فرآیندها باید از این موضوع اطمینان یابند که به طور همزمان داده ای را در shared memory نمی نویسند

SANA

Subject

Date. Month. Year.

۴) الف)



fork

fork

در اولین دور از حلقه علاوه بر فرآیند اجرا شده یک فرآیند فرزند هم تولید می‌شود که عیناً مانند فرآیند والد است آن فرزند و والد دوباره در حلقه بعدی هر کدام فرزندهایی به وجود می‌آورند بدیهی در آخرین دور از حلقه $2^{10} = 1024$ فرآیند خواهیم داشت

ب)

```
void main (){
    int i;
    for (i=0 ; i<10 ;i++){
        pid_t pid = fork();
        if (pid <0) {
            printf( "failded");
        } else if (pid ==0) {
            prnt( "child");
        }else {
            wait( NULL) ;
            printf ( "parent");
        }
    return 0;
}
```

SANA