

process

```
#include <sys/types.h>
#include <unistd.h>

pid_t pid = fork();

if (pid < 0)
{
    /* fork() failed */
}
else if (pid == 0)
{
    /* parent process */
}
else
{
    /* child process */
}
```

thread

```
/* remember to pass '-lpthread' to gcc */  
  
#include <pthread.h>  
  
void* thread_func(void *args)  
{  
    /* i run in my own thread */  
}  
  
int main()  
{  
    pthread_t thread;  
    pthread_create(&thread, NULL,  
                  thread_func, NULL);  
  
    /* do some work, maybe */  
  
    pthread_join(thread, NULL);  
}
```

problems

```
void* thread_func_a(void *args)
{
    printf("A");
    printf("B");
}
```

```
void* thread_func_b(void *args)
{
    printf("C");
    printf("D");
}
```

synchronization

```
/* global state */  
int a = 0;  
  
void* thread_func_a(void *args)  
{  
    a += 1  
}  
  
void* thread_func_b(void *args)  
{  
    a += 1;  
}
```

```
/* global state */
int a = 0;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void* thread_func_a(void *args)
{
    pthread_mutex_lock(&mutex);
    a += 1;
    pthread_mutex_unlock(&mutex);
}

void* thread_func_b(void *args)
{
    pthread_mutex_lock(&mutex);
    a += 1;
    pthread_mutex_unlock(&mutex);
}
```

caveats

```
void* thread_func_a(void *args)
{
    pthread_mutex_lock(&mutex_1);
    pthread_mutex_lock(&mutex_2);
    /* do some work */
    pthread_mutex_unlock(&mutex_2);
    pthread_mutex_unlock(&mutex_1);
}
```

```
void* thread_func_b(void *args)
{
    pthread_mutex_lock(&mutex_2);
    pthread_mutex_lock(&mutex_1);
    /* do some work */
    pthread_mutex_unlock(&mutex_1);
    pthread_mutex_unlock(&mutex_2);
}
```

context

```
int main()
{
    int sockfd = socket(...);
    bind(sockfd, ...);
    listen(sockfd, ...);

    while (1)
    {
        int clientfd = accept(sockfd, ...);

        while (1)
        {
            int bytes = recv(clientfd, ...);
            /* process client data */
            send(clientfd, ...);
        }
    }
}
```