threads

| idx | | idx | | idx | | idx | | idx | |
|---|---|---|---|---|---|---|---|---|---|
| function | arg | function | arg | function | arg | function | arg | function | arg |
| context | stack | context | stack | context | stack | context | stack | context | stack |
| Per_Dead | Rel_Per_D | Per_Dead | Rel_Per_D | Per_Dead | Rel_Per_D | Per_Dead | Rel_Per_D | Per_Dead | Rel_Per_D |
| | next | | next | | next | | next | | next |

Pointers

| |
|---|
| struct thread_block initp |
| thread freeQ |
| thread readyQ |
| thread doneQ |
| thread current |

```
spawn(computePower, 0);
spawn(computePower, 1);
computePrimes(2);
-----------------------------------
void spawn(void (* function)(int), int arg) {
    thread newp;
    DISABLE();
    if (!initialized)
        initializeThreads();
    newp = dequeue(&freeQ);
    newp->function = function;
    newp->arg = arg;
    newp->next = NULL;
    if (setjmp(newp->context) == 1) {
        ENABLE();
        current->function(current->arg);
        DISABLE();
        enqueue(current, &freeQ);
        dispatch(dequeue(&readyQ);
    }
    SETSTACK(&newp->context, &newp->stack);
    enqueue(newp, &readyQ);
    ENABLE();
}
void yield(void) {
    DISABLE();
    if (readyQ != NULL){
        thread p = dequeue(&readyQ);
        enqueue(current, &readyQ);
        dispatch(p);
    }
    ENABLE();
}
static void dispatch(thread next) {
    if (setjmp(current->context) == 0) {
        current = next;
        longjmp(next->context, 1);
    }
}
```

}