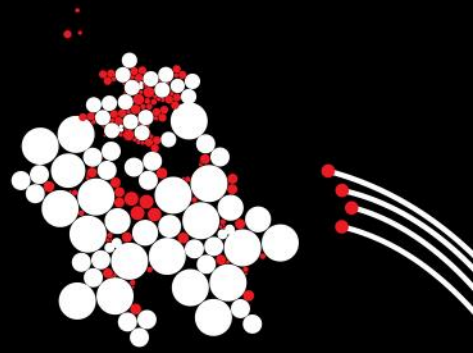
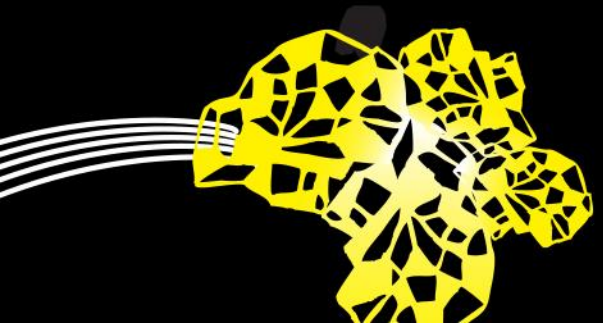
UNIVERSITY OF TWENTE.



OPERATING SYSTEMS

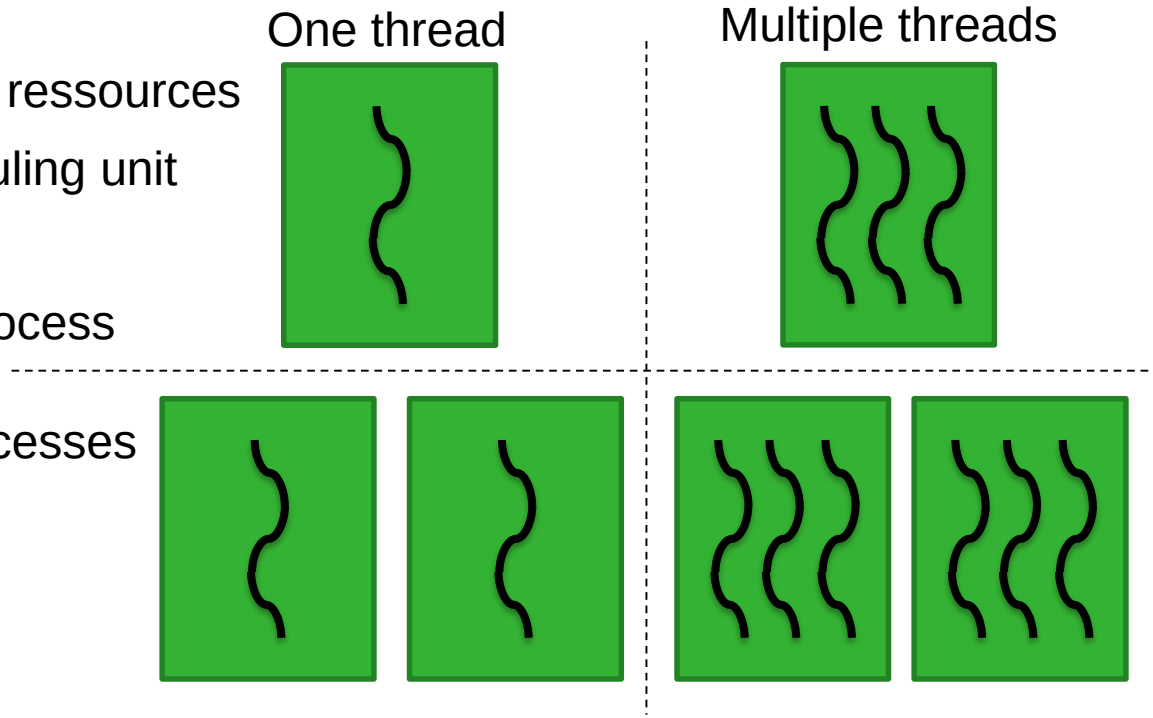
THREADS

ERIK TEWS <e.tews@utwente.nl>



PROCESSES VS. THREADS

- A process owns the ressources
- A thread is a scheduling unit



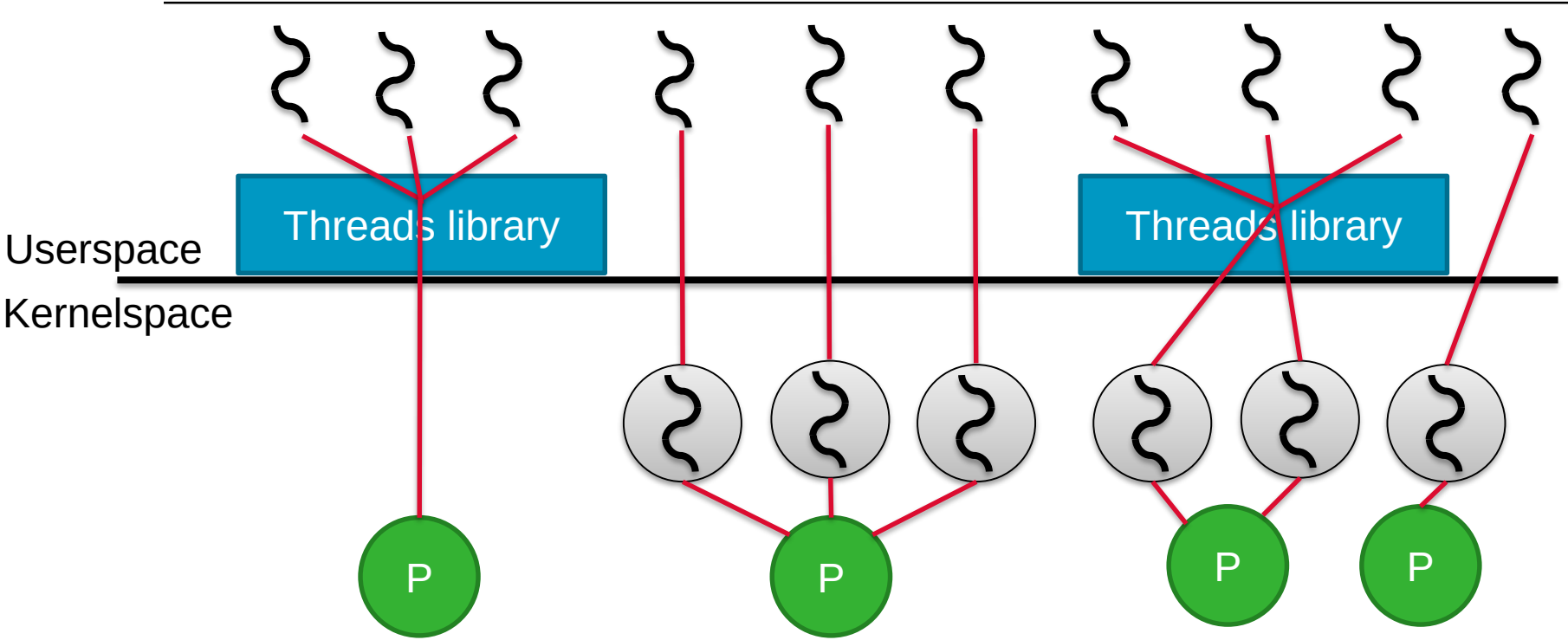
THREAD EXAMPLE

- Output?
- gcc MyThread.c –lpthread
- strace ./a.out
- Which system call?

```
void* tproc(void *arg) {
    printf("Thread %d\n", *((int *) arg));
    sleep(5);
    return NULL;
}
```

```
int main(int argc, char * argv[]) {
    int i;
    int targ[N];
    pthread_t tid[N];
    for(i = 0; i < N; i++) {
        targ[i] = i*i;
        if(pthread_create(&(tid[i]), NULL,
            &tproc, &targ[i]) != 0) {
            printf("Can't create %d\n", i);
            return 1;
        }
    }
    for(i = 0; i < N; i++) {
        if(pthread_join(tid[i], NULL) != 0) {
            printf("Can't join thread %d\n", i);
        }
    }
    return 0;
}
```

USER VS. KERNEL THREADS



DID YOU UNDERSTAND FORK AND THREADS?

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(int argc, char** argv) {
        int a = 0; pid_t pid = fork();
        if (pid < 0) { return 0; } // Fork failed
        if (pid == 0) { a++; // Child
        } else { // Parent, wait for the child
                wait(NULL);
                printf("%d\n", a);
        }
        return 0;
}
```

CONCURRENCY PROBLEMS

```c
typedef struct {
  int64_t x;
  int64_t y;
} coordinates;

const coordinates leftup = {.x = 1, .y = 1};
const coordinates rightdown = {.x = -1, .y = -1};
coordinates current;

void* tproc(void *arg) {
  int i = 1;
  while(1) {
    i = (i+1)&1;
    if (i == 0) {
      current = leftup;
    } else {
      current = rightdown;
    }
  }
}
```

```c
int main(int argc, char * argv[]) {
  coordinates r;
  pthread_t tid;
  current = leftup;
  if(pthread_create(&tid, NULL, &tproc, NULL) != 0) {
    printf("Can't create thread\n");
    return 1;
  }
  while (1) {
    r = current;
    if (r.x != r.y) {
      printf("wrong value: %"PRId64", %"PRId64"\n", r.x, r.y);
    }
  }
}
```

SUMMARY

- Threads interfere with the semantics of processes
- Linux treats threads as processes
- User level threads are more efficient than kernel level threads (or aren’t they)