50005Lab1

Implementation of main_loop() function

For each job with a corresponding `num` and `action`, the process is spinlocked into checking whether each process is alive or dead. I use a busy wait while-loop that is conditional on a Boolean variable `loop` with a nested iterative for-loop that cycles through the processes.

If a process is alive where `waitpid(children_processes[i], NULL, WNOHANG)==0` and also available to work `shmPTR_jobs_buffer[i].task_status==0`, then a job is assigned to the respective `shmPTR_jobs_buffer` shared memory buffer, and `loop` is set to `false` while a `break` statement is given to proceed to the next available job.

Otherwise, if the child is dead, the previous task is marked as completed, then the process is revived through a `fork()`. The child process is calls `job_dispatch(i)` with its respective process number `i`, while the parents assigns the process a new job and proceeds to the next job using the same procedure as above.

```
void main_loop(char *fileName) {
    FILE *opened_file = fopen(fileName, "r");
    char action; //stores whether its a 'p' or 'w'
    long num;    //stores the argument of the job
    while (fscanf(opened_file,"%c %ld\n", &action, &num)==2) { //while jobs available
        bool loop = true;
        while(loop) { //busy wait loop
            for ( int i = 0 ; i < number_of_processes ; i++ ) { //cycle through procs
                if (waitpid(children_processes[i], NULL, WNOHANG)==0) { //if alive
                    if (shmPTR_jobs_buffer[i].task_status==0) { //if job is done
                        shmPTR_jobs_buffer[i].task_type=action;
                        shmPTR_jobs_buffer[i].task_duration=num;
                        shmPTR_jobs_buffer[i].task_status=1;
                        sem_post(sem_jobs_buffer[i]);
                        loop = false; // break busy wait
                        break; //break for loop
                    }
                } else { //if child is dead
                    shmPTR_jobs_buffer[i].task_status=0;
                    int forkValue = fork();
                    children_processes[i]=forkValue;
                    if (forkValue < 0 ){ //if fork fails
                        perror("Failed to fork.\n" );
                        exit(1);
                    } else if(forkValue == 0) { //if child process
                        job_dispatch(i);
                    } else{ //parent job dispatch
                        shmPTR_jobs_buffer[i].task_status=0;
                        shmPTR_jobs_buffer[i].task_duration = num;
                        shmPTR_jobs_buffer[i].task_status = 1;
                        shmPTR_jobs_buffer[i].task_type = action;
                        sem_post(sem_jobs_buffer[i]); //post new job
                        loop = false;
                        break;
                    }
                }
            }
        }
    }
```