I familiarize myself with MPI_Pack function and setup an environment to test the replacement of memcpy.

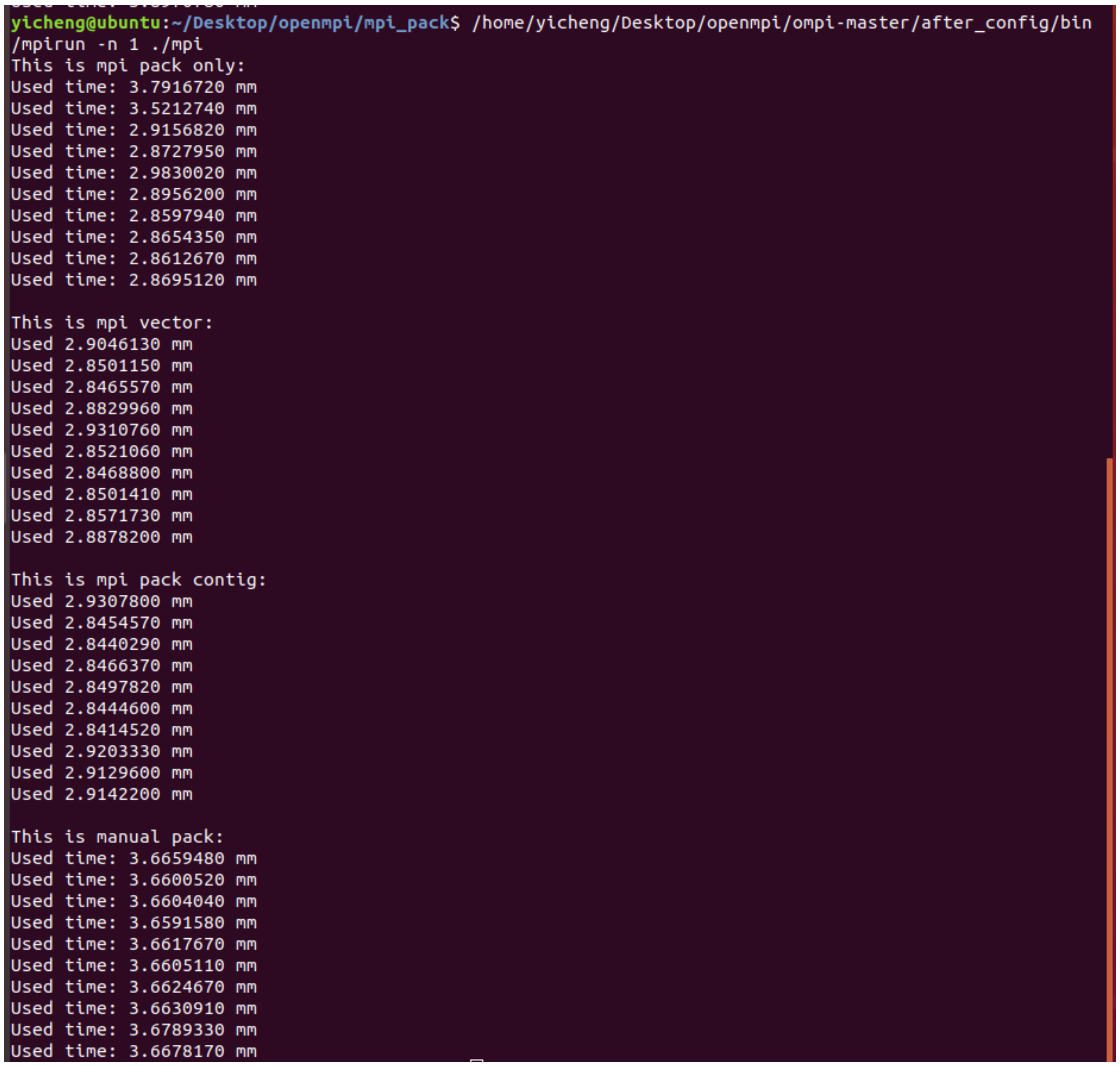
I'm packing 1,000,000 doubles and test the speed.

I used several structures for the packing function: MPI_contig, MPI_vector, MPI_contig.

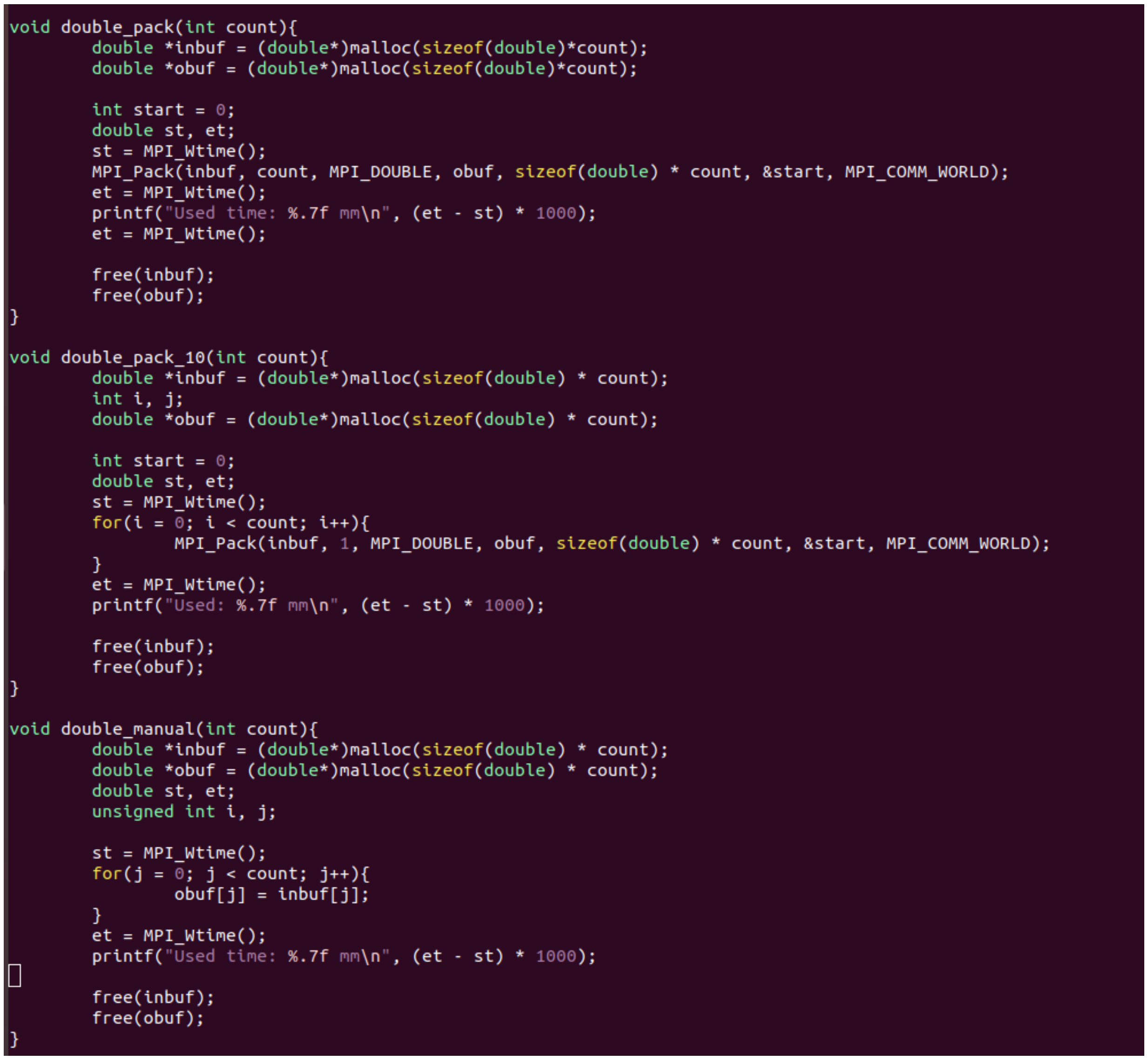
I'm comparing these structures with no structure packing (just MPI_Pack) and manually packing.

I'll start trying to replace memcpy function and try the speed for these tests.

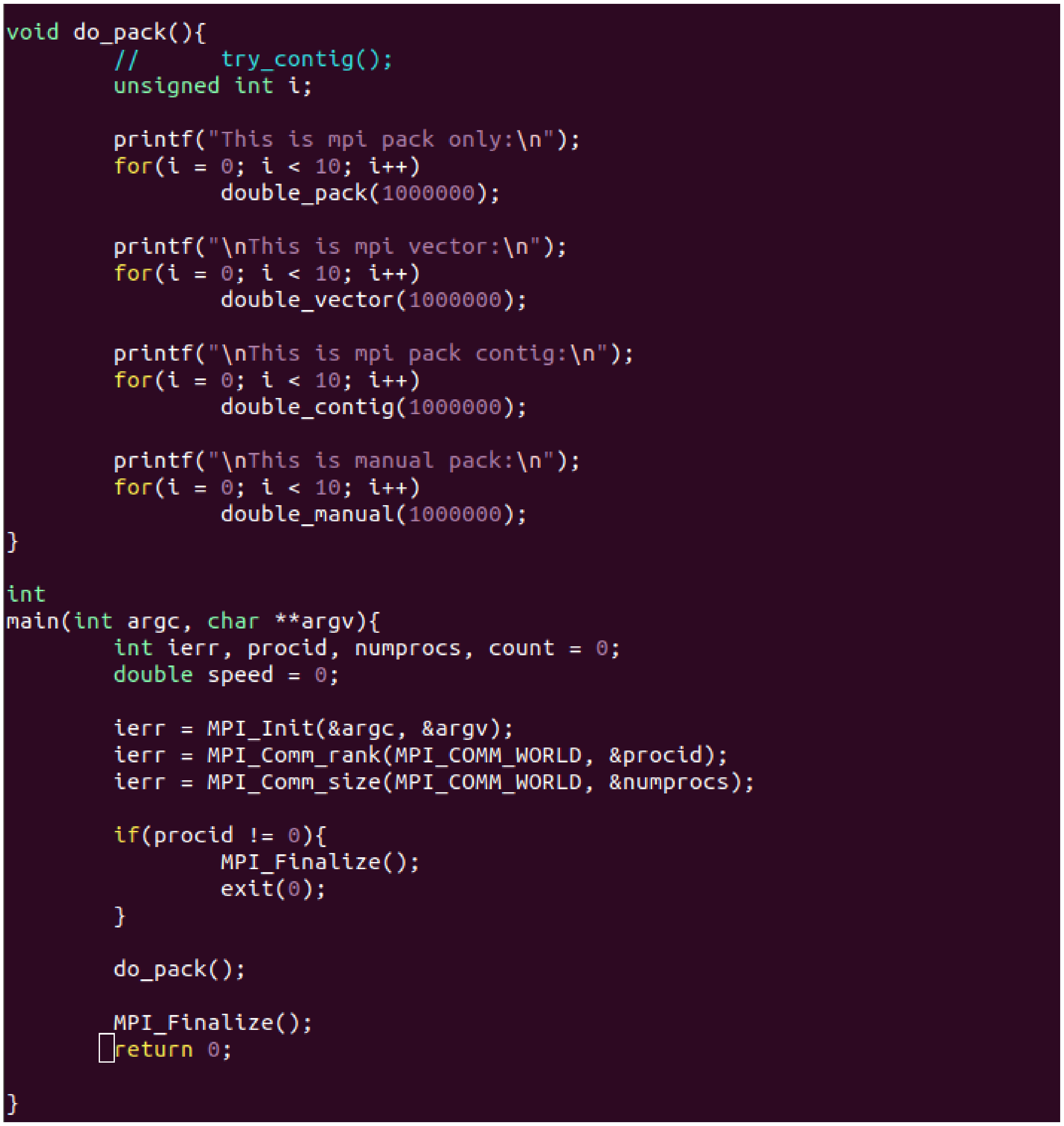
The results and functions are down below:



```
yicheng@ubuntu:~/Desktop/openmpi/mpi_pack$ /home/yicheng/Desktop/openmpi/ompi-master/after_config/bin
/mpirun -n 1 ./mpi
This is mpi pack only:
Used time: 3.7916720 mm
Used time: 3.5212740 mm
Used time: 2.9156820 mm
Used time: 2.8727950 mm
Used time: 2.9830020 mm
Used time: 2.8956200 mm
Used time: 2.8597940 mm
Used time: 2.8654350 mm
Used time: 2.8612670 mm
Used time: 2.8695120 mm

This is mpi vector:
Used 2.9046130 mm
Used 2.8501150 mm
Used 2.8465570 mm
Used 2.8829960 mm
Used 2.9310760 mm
Used 2.8521060 mm
Used 2.8468800 mm
Used 2.8501410 mm
Used 2.8571730 mm
Used 2.8878200 mm

This is mpi pack contig:
Used 2.9307800 mm
Used 2.8454570 mm
Used 2.8440290 mm
Used 2.8466370 mm
Used 2.8497820 mm
Used 2.8444600 mm
Used 2.8414520 mm
Used 2.9203330 mm
Used 2.9129600 mm
Used 2.9142200 mm

This is manual pack:
Used time: 3.6659480 mm
Used time: 3.6600520 mm
Used time: 3.6604040 mm
Used time: 3.6591580 mm
Used time: 3.6617670 mm
Used time: 3.6605110 mm
Used time: 3.6624670 mm
Used time: 3.6630910 mm
Used time: 3.6789330 mm
Used time: 3.6678170 mm
```



```c
void double_pack(int count){
        double *inbuf = (double*)malloc(sizeof(double)*count);
        double *obuf = (double*)malloc(sizeof(double)*count);

        int start = 0;
        double st, et;
        st = MPI_Wtime();
        MPI_Pack(inbuf, count, MPI_DOUBLE, obuf, sizeof(double) * count, &start, MPI_COMM_WORLD);
        et = MPI_Wtime();
        printf("Used time: %.7f mm\n", (et - st) * 1000);
        et = MPI_Wtime();

        free(inbuf);
        free(obuf);
}

void double_pack_10(int count){
        double *inbuf = (double*)malloc(sizeof(double) * count);
        int i, j;
        double *obuf = (double*)malloc(sizeof(double) * count);

        int start = 0;
        double st, et;
        st = MPI_Wtime();
        for(i = 0; i < count; i++){
                MPI_Pack(inbuf, 1, MPI_DOUBLE, obuf, sizeof(double) * count, &start, MPI_COMM_WORLD);
        }
        et = MPI_Wtime();
        printf("Used: %.7f mm\n", (et - st) * 1000);

        free(inbuf);
        free(obuf);
}

void double_manual(int count){
        double *inbuf = (double*)malloc(sizeof(double) * count);
        double *obuf = (double*)malloc(sizeof(double) * count);
        double st, et;
        unsigned int i, j;

        st = MPI_Wtime();
        for(j = 0; j < count; j++){
                obuf[j] = inbuf[j];
        }
        et = MPI_Wtime();
        printf("Used time: %.7f mm\n", (et - st) * 1000);

        free(inbuf);
        free(obuf);
}
```



```
void do_pack(){
        //      try_contig();
        unsigned int i;

        printf("This is mpi pack only:\n");
        for(i = 0; i < 10; i++)
                double_pack(1000000);

        printf("\nThis is mpi vector:\n");
        for(i = 0; i < 10; i++)
                double_vector(1000000);

        printf("\nThis is mpi pack contig:\n");
        for(i = 0; i < 10; i++)
                double_contig(1000000);

        printf("\nThis is manual pack:\n");
        for(i = 0; i < 10; i++)
                double_manual(1000000);
}

int
main(int argc, char **argv){
        int ierr, procid, numprocs, count = 0;
        double speed = 0;

        ierr = MPI_Init(&argc, &argv);
        ierr = MPI_Comm_rank(MPI_COMM_WORLD, &procid);
        ierr = MPI_Comm_size(MPI_COMM_WORLD, &numprocs);

        if(procid != 0){
                MPI_Finalize();
                exit(0);
        }

        do_pack();

        MPI_Finalize();
        return 0;

}
```