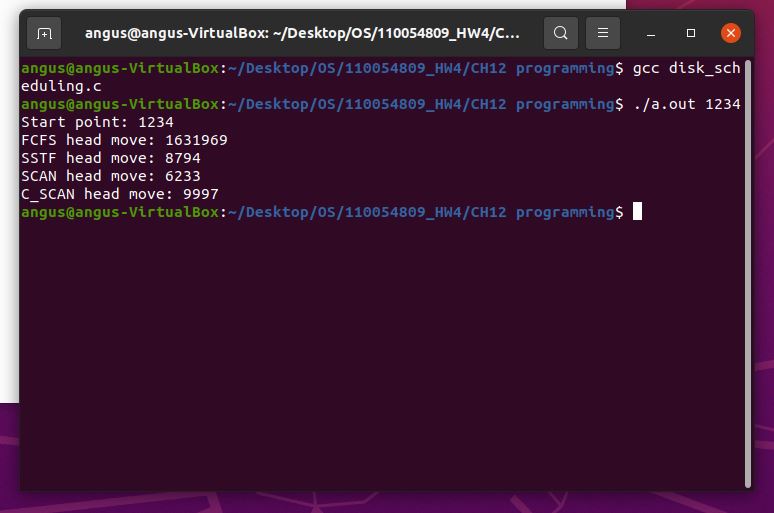
Steps：

1. gcc disk\_scheduling.c
2. ./a.out 1234

Snapshot：



Source code：

#include<stdio.h>

#include<stdlib.h>

#define CYLINDERS 5000

#define REQUESTS 1000

int start;

int searchAry[REQUESTS];

int cmp(const void \*a, const void \*b) {

return \*(int \*)a - \*(int \*)b;

}

int FCFS(int \*ary) {

int result = 0, currentHead = start;

for (size\_t i = 0; i < REQUESTS; i++) {

result += abs(currentHead - ary[i]);

currentHead = ary[i];

}

return result;

}

int findIdx(int \*ary) {

int left = 0, right = REQUESTS;

while(left <= right) {

int mid = (left + right) / 2;

if (ary[mid] > start) {

right = mid - 1;

} else if (ary[mid] < start) {

left = mid + 1;

} else {

return mid;

}

}

return left;

}

int SSTF(int \*ary) {

qsort(ary, REQUESTS, sizeof(int), cmp);

// for (size\_t i = 0; i < REQUESTS; i++) {

// printf("%d ", ary[i]);

// }

// printf("\n");

int result = 0;

int smallIdx = findIdx(ary)-1, largeIdx = findIdx(ary);

int smallDif = 0, largeDif = 0, currentHead = start;

for (size\_t i = 0; i < REQUESTS; i++) {

// printf("%d \t", currentHead);

smallDif = abs(currentHead - ary[smallIdx]);

largeDif = abs(currentHead - ary[largeIdx]);

if (smallDif < largeDif) {

result += smallDif;

currentHead = ary[smallIdx];

smallIdx--;

} else {

result += largeDif;

currentHead = ary[largeIdx];

largeIdx++;

}

}

return result;

}

int SCAN(int \*ary) {

qsort(ary, REQUESTS, sizeof(int), cmp);

int result = 0;

int index = findIdx(ary)-1;

int currentHead = start;

//down

for (int i = index; i >= 0; i--) {

result += abs(currentHead - ary[i]);

currentHead = ary[i];

}

result += abs(ary[0]);

currentHead = 0;

//up

for (int i = index; i < REQUESTS; i++) {

result += abs(currentHead - ary[i]);

currentHead = ary[i];

}

return result;

}

int C\_SCAN(int \*ary) {

qsort(ary, REQUESTS, sizeof(int), cmp);

int result = 0;

int index = findIdx(ary);

int currentHead = start;

//up

for (size\_t i = index; i < REQUESTS; i++) {

result += abs(ary[i] - currentHead);

currentHead = ary[i];

}

result += CYLINDERS - 1 - ary[REQUESTS-1];

result += CYLINDERS - 1;

currentHead = 0;

//0~index

for (size\_t i = 0; i < index; i++) {

result += abs(ary[i] - currentHead);

currentHead = ary[i];

}

return result;

}

int main(int argc, char \*argv[]) {

start = atoi(argv[1]);

if(argc != 2) {

printf("Need to enter a starting point between 0-4999. Ex. ./disk\_scheduling 1000\n");

exit(-1);

}

for (size\_t i = 0; i < REQUESTS; i++) {

searchAry[i] = rand() % CYLINDERS;

}

// start = 2150;

// int testAry[] = {2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681};

printf("Start point: %d\n", start);

printf("FCFS head move: %d\n", FCFS(searchAry));

printf("SSTF head move: %d\n", SSTF(searchAry));

printf("SCAN head move: %d\n", SCAN(searchAry));

printf("C\_SCAN head move: %d\n", C\_SCAN(searchAry));

return 0;

}