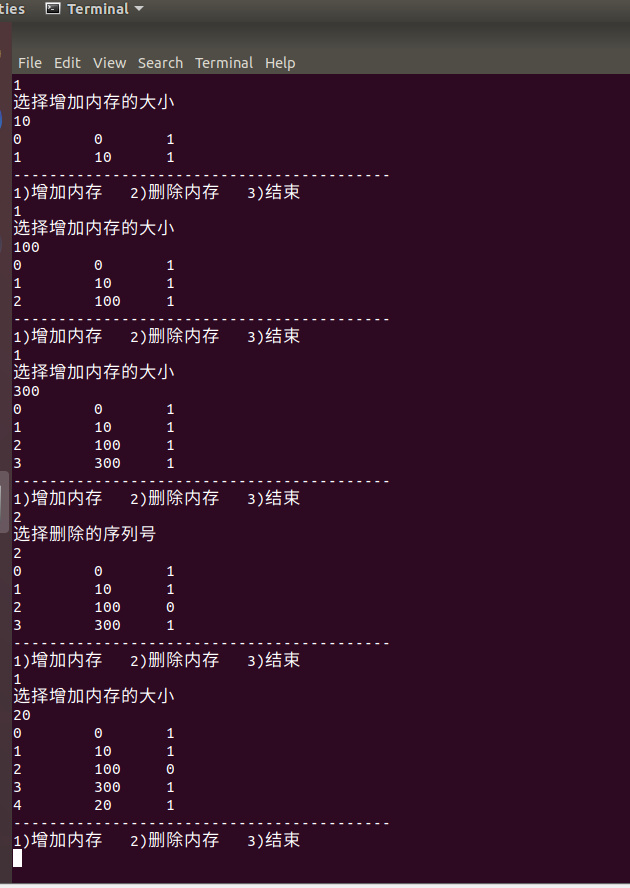
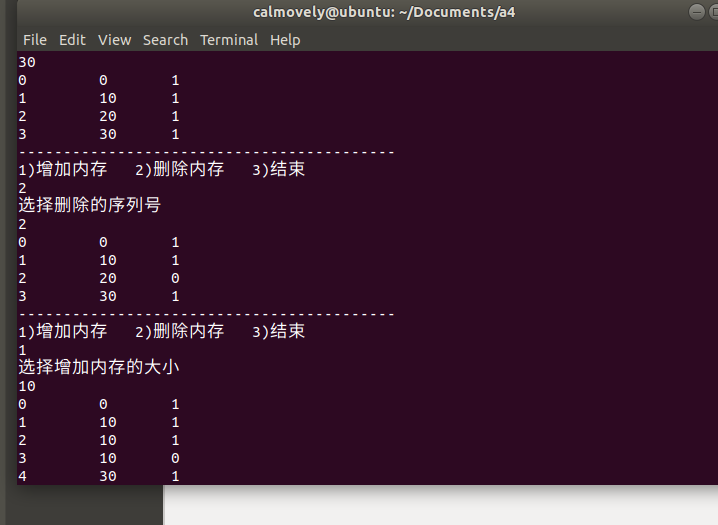
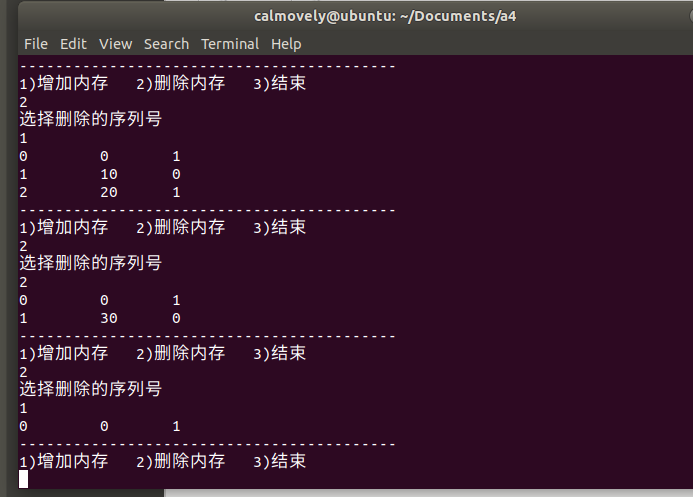
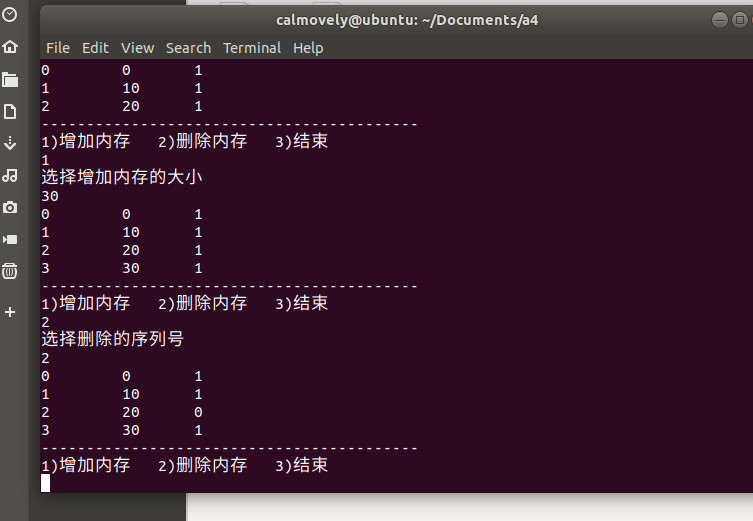
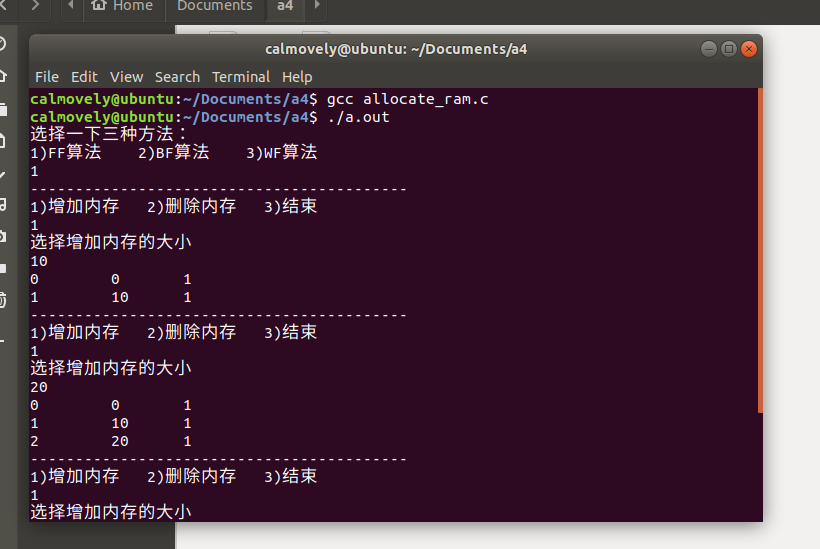
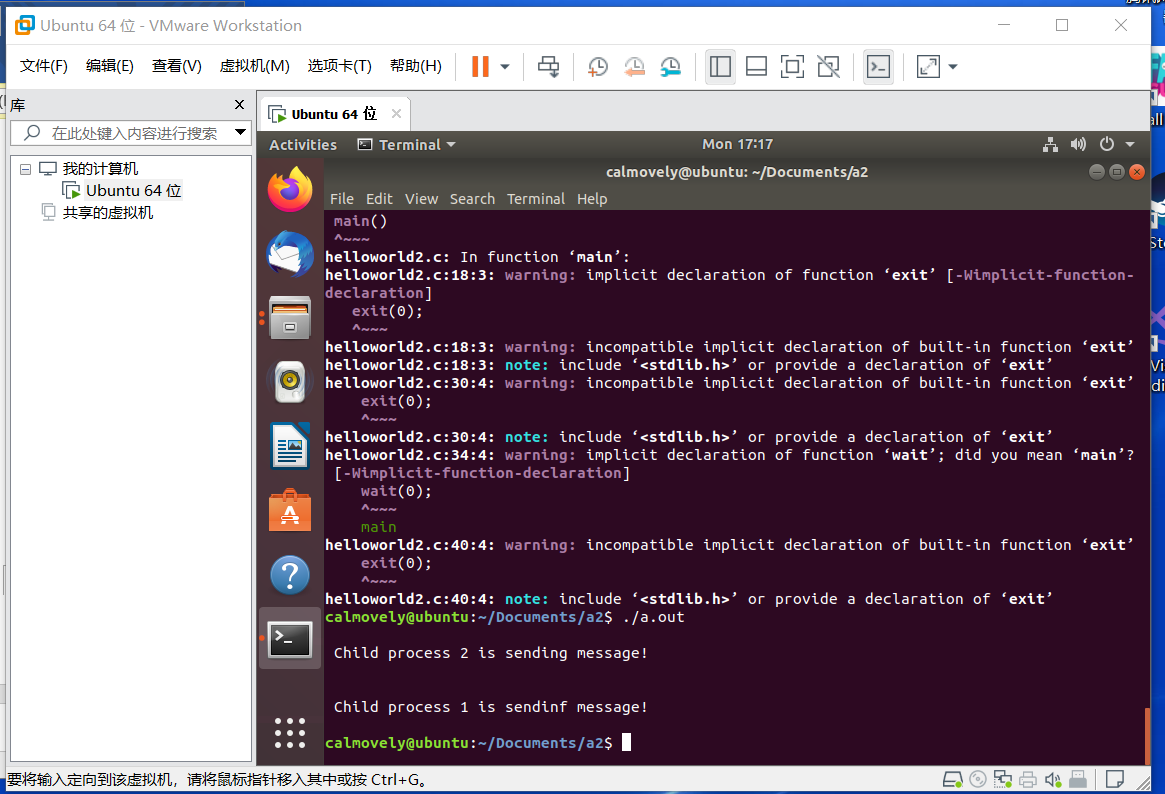
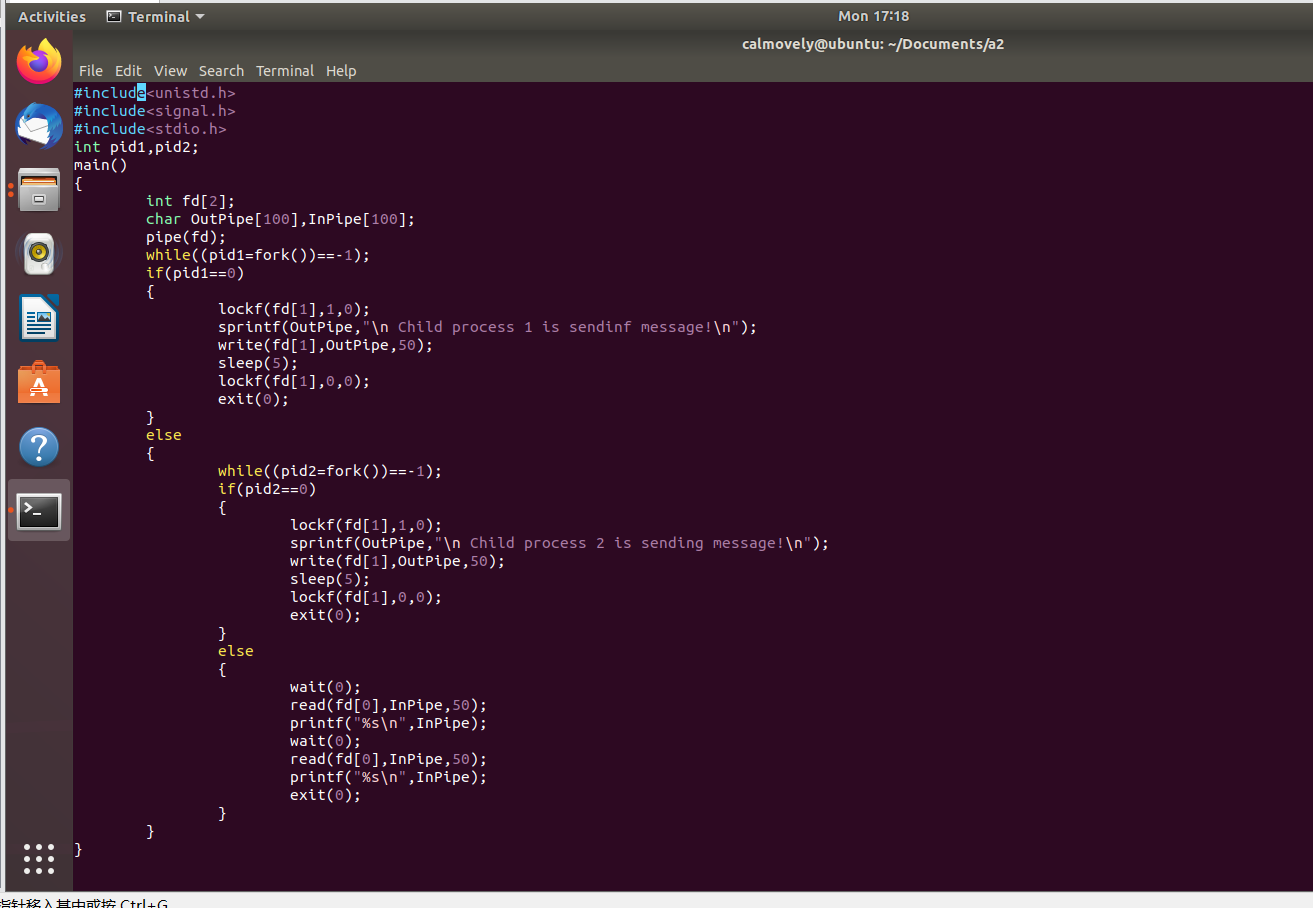
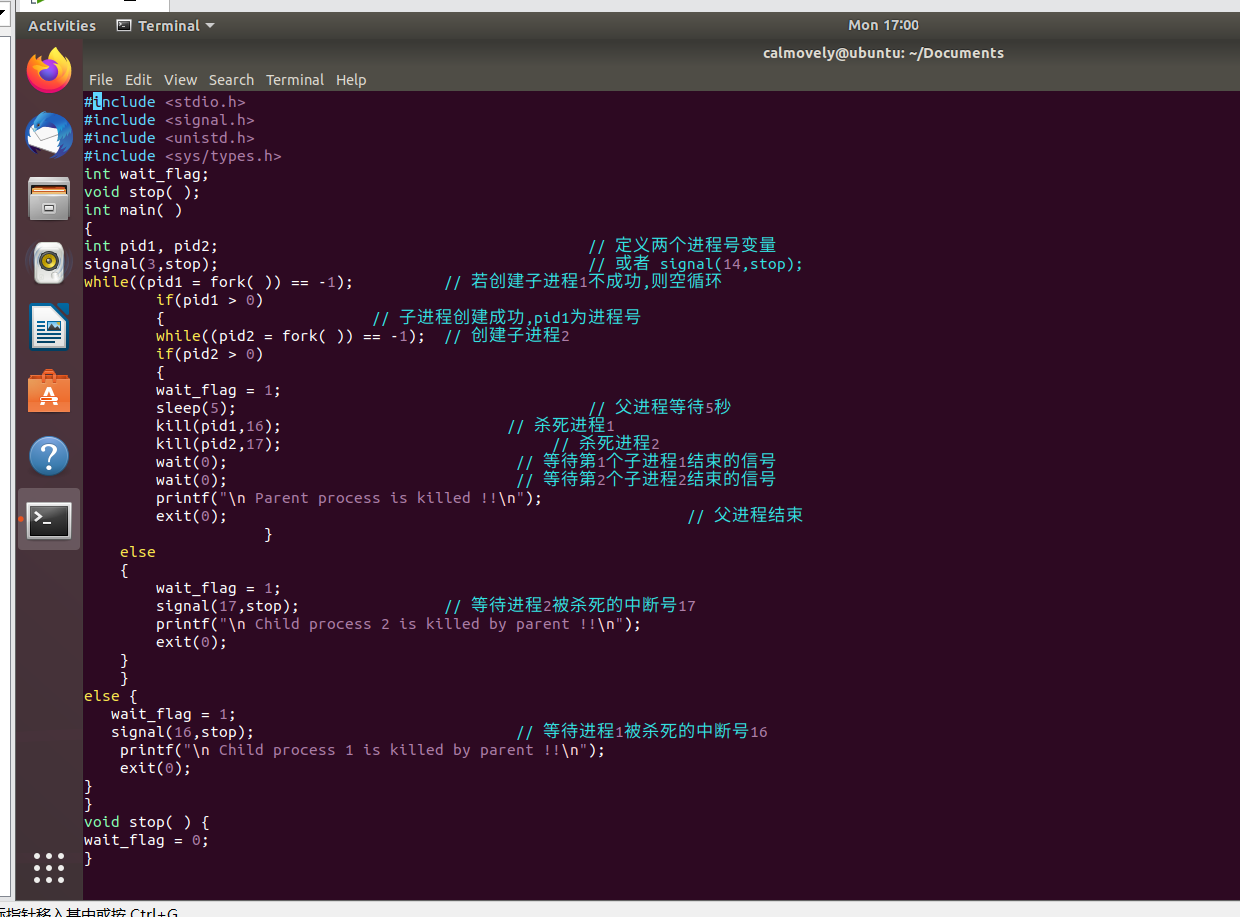
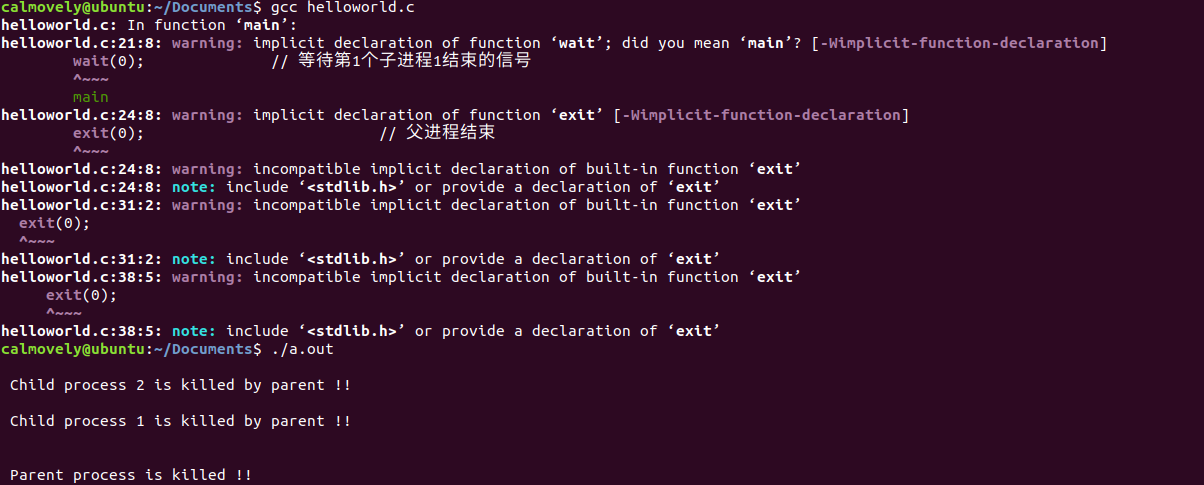
~/Document/



代码：

#include<stdio.h>

#include<stdlib.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

#define max\_mem 2048;

typedef struct list

{

struct list \*next;

int size;

struct list \*pre;

int state;

int addr;

}list;

struct list \*head;

struct list \*tail;

void initial()

{

head = (struct list\*)malloc(sizeof(struct list));

tail = (struct list\*)malloc(sizeof(struct list));

head->pre = NULL;

head->next = tail;

tail->pre = head;

tail->next = NULL;

head->state = 1;

tail->state = 0;

head->addr = 0;

tail->addr = 1;

head->size = 0;

tail->size = max\_mem;

}

void print()

{

struct list \*p = head;

while (p != tail)

{

printf("%d \t %d \t %d\n", p->addr, p->size, p->state);

p = p->next;

}

}

void ff(int res)

{

struct list \*p = head;

while (p)

{

if (p->state == 0 && p->size == res)

{

p->state = 1;

return;

}

else if (p->state == 0 && p->size > res)

{

struct list \*t = (struct list\*)malloc(sizeof(struct list));

t->pre = p->pre;

t->next = p;

p->pre->next = t;

p->pre = t;

t->size = res;

p->size -= res;

t->addr = p->addr;

t->state = 1;

while (p)

{

p->addr++;

p = p->next;

}

return;

}

else

{

p = p->next;

}

}

printf("Error:out of range!");

}

void bf(int res)

{

int ch;

struct list \*temp = (struct list\*)malloc(sizeof(struct list));

temp->size = res;

temp->state = 1;

struct list \*p = head->next;

struct list \*q = NULL;

while (p)

{

if (p->state == 0 && (p->size >= res))

{

if (q == NULL)

{

q = p;

ch = p->size - res;

}

else if (q->size > p->size)

{

q = p;

ch = p->size - res;

}

}

p = p->next;

}

if (q == NULL)

{

return;

}

else if (q->size == res)

{

q->state = 1;

}

else

{

temp->pre = q->pre;

temp->next = q;

temp->addr = q->addr;

q->pre->next = temp;

q->pre = temp;

q->size = ch;

while (q)

{

q->addr += 1;

q = q->next;

}

return;

}

printf("Rrror: out of range!\n");

return;

}

//最差适应算法

void wf(int res)

{

int ch; //记录最大剩余空间

struct list \*temp = (struct list\*)malloc(sizeof(struct list));

temp->size = res;

temp->state = 1;

list \*p = head->next;

list \*q = NULL; //记录最佳插入位置

while (p) //初始化最大空间和最佳位置

{

if (p->state == 0 && (p->size >= res))

{

if (q == NULL)

{

q = p;

ch = p->size - res;

}

else if (q->size < p->size)

{

q = p;

ch = p->size - res;

}

}

p = p->next;

}

if (q == NULL) return;//没有找到空闲块

else if (q->size == res)

{

q->state = 1;

return;

}

else

{

temp->pre = q->pre;

temp->next = q;

temp->addr = q->addr;

q->pre->next = temp;

q->pre = temp;

q->size = ch;

while (q)

{

q->addr += 1;

q = q->next;

}

return;

}

printf("Error: out of range!\n");

return;

}

void myremove(int n)

{

struct list \*p = head;

while (p->addr != n)

{

p = p->next;

}

p->state = 0;

if (p->next == tail)

{

p->pre->next = tail;

tail->pre = p->pre;

tail->addr--;

}

if (p->pre->state == 0)

{

p->size += p->pre->size;

p->pre = p->pre->pre;

p->pre->next = p;

p->addr--;

}

if (p->next->state == 0 && p->next != tail)

{

p->size += p->next->size;

p->next = p->next->next;

p->next->pre = p;

p = p->next;

while (p)

{

p->addr--;

p = p->next;

}

}

}

void myc(int c, int res)

{

switch (c)

{

case 1:

ff(res); break;

case 2:

bf(res); break;

case 3:

wf(res); break;

default:

break;

}

}

int main()

{

initial();

int choice;

int res = 0;

int c = 0;

printf("选择一下三种方法：\n1)FF算法 2)BF算法 3)WF算法\n");

scanf("%d", &c);

while (1)

{

printf("------------------------------------------\n");

printf("1)增加内存 2)删除内存 3)结束\n");

scanf("%d", &choice);

switch (choice)

{

case 1:

printf("选择增加内存的大小\n");

scanf("%d", &res);

myc(c, res);

print();

break;

case 2:

printf("选择删除的序列号\n");

scanf("%d", &res);

myremove(res);

print();

break;

case 3:

return 0;

break;

default:

break;

}

}

printf("Program Exit.\n");

return 0;

}