

8.

执行结果

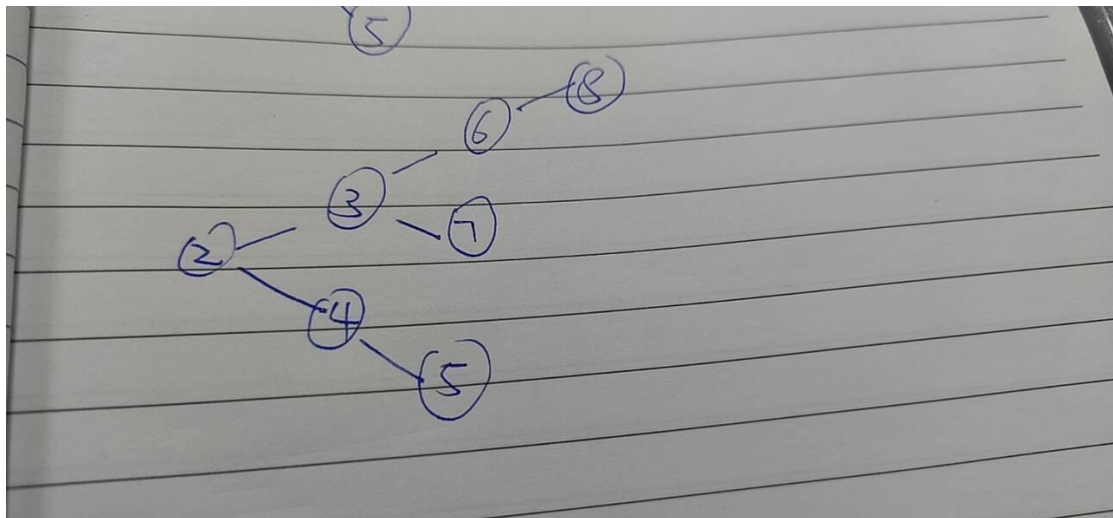
Level 2

Level 4

Level 2

Level 3

Level 2



9.

CHILD: 0

CHILD: -1

CHILD: -2

CHILD: -3

CHILD: -4

PARENT: 0

PARENT: 0

PARENT: 1

PARENT: 2

PARENT: 3

PARENT: 4

Wait 函数的作用：阻塞父进程，让父进程等待其子进程结束

13.

```
#include <stdio.h>
```

```
#include <signal.h>
```

```
#include <sys/time.h>
```

```
#include <string.h>
```

```
#include <unistd.h>
```

```
#include <stdlib.h>
```

```

volatile sig_atomic_t stop = 0;

void handle_alarm(int sig) {
    printf("Alarm!\n");
}

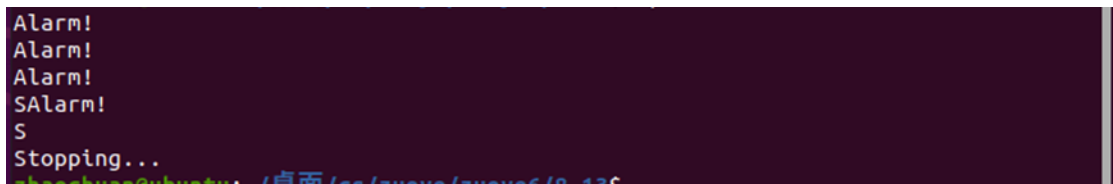
int main() {
    struct sigaction sa;
    memset(&sa, 0, sizeof(sa));
    sa.sa_handler = handle_alarm;
    sigaction(SIGALRM, &sa, NULL);

    struct itimerval timer;
    timer.it_value.tv_sec = 3;
    timer.it_value.tv_usec = 0;
    timer.it_interval.tv_sec = 3;
    timer.it_interval.tv_usec = 0;
    setitimer(ITIMER_REAL, &timer, NULL);

    char input[2];
    while (1) {
        if (fgets(input, sizeof(input), stdin) != NULL) {
            if (input[0] == 'S' || input[0] == 's') {
                printf("Stopping...\n");
                timer.it_value.tv_sec = 0;
                timer.it_value.tv_usec = 0;
                timer.it_interval.tv_sec = 0;
                timer.it_interval.tv_usec = 0;
                setitimer(ITIMER_REAL, &timer, NULL);
                break;
            }
        }
    }

    return 0;
}

```



```

Alarm!
Alarm!
Alarm!
SAlarm!
S
Stopping...

```

6.

虚拟地址空间大小：32 位系统虚拟地址空间为 2^{32} B

页大小：2KB= 2^{11} B

虚页数量： 2^{21}

虚页页号：0x0030f40（需补全为 32 位地址 0x00030f40）

二进制表示：

00000000000000000000000000000000

页号 (VPN)：取高 21 位（右移 11 位）：

00000000000000000000000000000001 → 十进制值=97.

页内偏移：取低 11 位：

11101000000 → 十六进制=0x740（十进制=1,856）

9.

选择（2）具有 20GB 物理主存的 x86-64 机器。

原因：（1）的 PAE 方案虚拟地址空间：32 位 → 每个进程最多使用 4GB 虚拟内存，如果高并发大量进程共享内存，则限制较大。（2）的虚拟内存足够大，虽然物理内存较小（20GB），但可通过优化内存管理和换页机制缓解，而 x86-32 的 4GB 虚拟地址空间是硬性瓶颈，无法通过 PAE 解决

10.

FIFO

访问序列	缓存状态（队列顺序）	缺页次数	备注
2	[2]	1	缺页
3	[2, 3]	2	缺页
2	[2, 3]	2	命中
1	[2, 3, 1]	3	缺页（缓存未满）
5	[3, 1, 5]	4	缺页，替换最早进入的 2
2	[1, 5, 2]	5	缺页，替换最早进入的 3
4	[5, 2, 4]	6	缺页，替换最早进入的 1
5	[5, 2, 4]	6	命中
3	[2, 4, 3]	7	缺页，替换最早进入的 5
2	[2, 4, 3]	7	命中
5	[4, 3, 5]	8	缺页，替换最早进入的 2
2	[3, 5, 2]	9	缺页，替换最早进入的 4

FIFO缺页次数：9次。

LRU

访问序列	缓存状态（最近访问顺序）	缺页次数	备注
2	[2]	1	缺页
3	[2, 3]	2	缺页
2	[3, 2]	2	命中，更新顺序
1	[3, 2, 1]	3	缺页（缓存未满）
5	[2, 1, 5]	4	缺页，替换最久未使用的3
2	[1, 5, 2]	4	命中，更新顺序
4	[5, 2, 4]	5	缺页，替换最久未使用的1
5	[2, 4, 5]	5	命中，更新顺序
3	[4, 5, 3]	6	缺页，替换最久未使用的2
2	[5, 3, 2]	7	缺页，替换最久未使用的4
5	[3, 2, 5]	7	命中，更新顺序
2	[3, 5, 2]	7	命中，更新顺序

LRU缺页次数：7次。

11

7.

copy_array(a + 1, a, 999)

操作：将 a[1] 到 a[999] 复制到 a[0] 到 a[998]

copy_array(a, a + 1, 999)

操作：将 a[0] 到 a[998] 复制到 a[1] 到 a[999]

原因：现代处理器采用流水线，第一个写入操作不影响后续读取，减少流水线停顿，而第二个的写入操作会影响后续的读取