

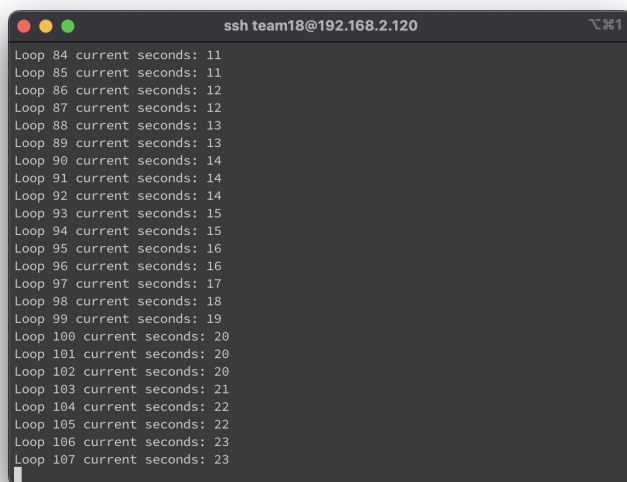
Midterm 2

Team18

112062519 廖思愷 112062636 游竣量 111065547 游述宇

1) (2.5%) Write a program that calls `sleep(10)` in an infinite loop. Every six times through the loop (every minute), fetch the current time of day and print the `tm_sec` field. Run the program for 3+ hours and explain the results. How would a program such as the cron daemon, which runs every minute on the minute, handle this situation? Please answer the question and write your observations in the report.

make 後，執行 `./q1.c` 即可每分鐘印出當前秒數：

A terminal window titled 'ssh team18@192.168.2.120' showing the output of a program. The output consists of multiple lines, each starting with 'Loop' followed by a number and 'current seconds:'. The seconds values are 11, 11, 12, 12, 13, 13, 14, 14, 15, 15, 16, 16, 17, 18, 19, 20, 20, 20, 21, 21, 22, 22, 23, 23. This indicates that the program prints the current seconds every 10 seconds, and the values increase by 1 every 6 seconds (10 seconds / 6 loops = 1.66 seconds per loop, rounded to 2 seconds per loop for the first 5 loops, then 1 second for the 6th loop, and so on).

```
ssh team18@192.168.2.120
Loop 84 current seconds: 11
Loop 85 current seconds: 11
Loop 86 current seconds: 12
Loop 87 current seconds: 12
Loop 88 current seconds: 13
Loop 89 current seconds: 13
Loop 90 current seconds: 14
Loop 91 current seconds: 14
Loop 92 current seconds: 15
Loop 93 current seconds: 15
Loop 94 current seconds: 16
Loop 95 current seconds: 16
Loop 96 current seconds: 17
Loop 97 current seconds: 17
Loop 98 current seconds: 18
Loop 99 current seconds: 18
Loop 100 current seconds: 19
Loop 101 current seconds: 19
Loop 102 current seconds: 20
Loop 103 current seconds: 20
Loop 104 current seconds: 20
Loop 105 current seconds: 21
Loop 106 current seconds: 21
Loop 107 current seconds: 22
Loop 108 current seconds: 22
Loop 109 current seconds: 23
Loop 110 current seconds: 23
```

由每分鐘打印出的當前秒數可以觀察到，秒數時間並不精確同步，預期的輸出結果應該會跟開始時間秒數同秒，實際秒數卻是有往後延遲，逐漸偏離開始時間秒數。

cron 是一個在 **Unix-like** 系統中用於時間基礎的作業調度的守護進程，它可以準確地在指定時間執行任務，**cron** 是根據系統的實際時間來觸發任務，不是依賴於 **sleep** 函數的固定延遲，所以即使系統負載較高或有輕微的執行延遲，**cron** 仍然可以在每分鐘的開始時刻準確地觸發任務。

cron daemon 是透過一個稱為 **crontab** 的配置文件，用戶可以編輯自己的 **crontab** 文件來安排任務，然後 **cron** 會定期檢查此文件，並在系統時間符合某個任務指定的時間時，啟動該任務。

2) (2.5%) Write a program to find all the symbolic links under a given path, and print the content of the symbolic links (actual pathnames). You may assume the pathnames are no longer than 512 bytes.

make 後，執行 **./q2 <pathname>**，將 **<pathname>** 換為任何指定的路徑，即可印出該路徑下所有 **symbolic links** 的 **actual pathnames**。

example:

```
./q2 /
```

```
./q2 ./test_q2
```

執行結果：

```
team18@:~/Kyle_test/midterm2 $ ./q2 ./test_q2
"./test_q2/link_to_file1" -> "file1.txt"
"./test_q2/link_to_file2" -> "file2.txt"
"./test_q2/link_to_hosts" -> "/etc/hosts"
"./test_q2/dir1/dir2/link_to_file3" -> "file3.txt"
"./test_q2/dir1/dir2/link_to_file4" -> "file4.txt"
```

3) (2.5%) Write a utility to print the current time in the following format: “Hour:Min:Sec, Week day Month Date, Year” (24-hour format, full month name). For example, 10:35:03, Tuesday November 29, 2016.

make 後，執行 **./q3.c** 即可印出當前時間：

```
team18@:~/Advanced-UNIX-Programming_team18/midterm2 $ cat q3.c
#include <stdio.h>
#include <time.h>

int main() {
    time_t rawtime;
    struct tm *currenttime;
    char buffer[80];

    time(&rawtime);
    currenttime = localtime(&rawtime);

    strftime(buffer, sizeof(buffer), "%H:%M:%S, %A %B %d, %Y", currenttime);
    printf("%s\n", buffer);

    return 0;
}team18@:~/Advanced-UNIX-Programming_team18/midterm2 $ ./q3
19:44:04, Thursday December 07, 2023
```

4) (2.5%) A sample code (program.c) converts an integer array into a link list and sort the given list in the ascending order. However, there are some bugs in the source code (program.c), please fix the bugs and write your implementation in the report. Your submitted q4.c shall produce correct answers, without any compilation/run-time errors and warning, including memory leaks. Notice that you can not add new functions nor change the signature (say parameters) of existing functions.

在 **append_node** 函數中用 **malloc** 分配記憶體：

```
struct ListNode *append_node(struct ListNode *head, int val) {
    struct ListNode *newNode = (struct ListNode *)malloc(sizeof(struct ListNode));

    newNode->val = val;
    newNode->next = NULL;

    if (head == NULL) {
        return newNode;
    } else {
        struct ListNode *tail = head;
        while (tail->next != NULL)
            tail = tail->next;

        tail->next = newNode;
    }

    return head;
}
```

而在程式執行結束前再使用 **free** 釋放記憶體：

```
struct ListNode *tmp;
while (head != NULL) {
    {
        tmp = head;
        head = head->next;
        free(tmp);
    }
}
```

make 後，執行 **./q4.c** 即可輸出結果：

```
team18@~/Advanced-UNIX-Programming_team18/midterm2 $ ./q4
1
2
3
4
5
6
```