Lab8實作流程：

1. 首先我們知道原始檔中含有
   * 1-01/ 1-02 / … / 1-52

* 2-01 / 2-02 / … / 2-52
* 3-01 / 3-02 / … / 3-52
* 4-01 / 4-02 / … / 4-52
* 5-01 / 5-02 / … / 5-52

合計52\*5個txt檔

1. 接著我們要在我們的父程序中產生**五個子程序:**
   * + - 第一個子程序負責1-01到1-52的txt檔
       - 第二個子程序負責2-01到2-52的txt檔
       - 第三個子程序負責3-01到3-52的txt檔
       - 第四個子程序負責4-01到4-52的txt檔
       - 第五個子程序負責5-01到5-52的txt檔

/\* 可以用for實現, 且五個子程序同時運行 \*/

1. 然後讓每個子程序依序打開自己負責01到52的txt檔

/\* 可以用for + sprintf() + fopen() 實現 \*/

1. 當每次打開一個txt檔時, 依序對每天的96個用電量做加總後存入sum.txt /\* 直接呼叫void accumulation() \*/
2. 所以一個子程序總共會呼叫52\*7次的accumulation(), 有五個

子程序所以accumulation()總共會被呼叫52\*7\*5次, **如果有**

**兩個以上的子程序同時去呼叫accumulation()就有機會發生**

**Race condition,** 所以我們要避免這種情況

1. 如何避免 → 利用以下函式 (定義在tellwait.c內)
   * TELL\_WAIT(void)
   * TELL\_PARENT(pid\_t pid)
   * WAIT\_PARENT(void)
   * TELL\_CHILD(pid\_t pid)
   * WAIT\_CHILD(void)
2. 子程序呼叫accumulation()前先等待父程序, 等父程序說可以時

候才呼叫accumulation(), 而程式中有五個子程序所以只有最

快那個可以先呼叫, 其他子程序要等到被父程序告知後才可

以去呼叫accumulation()

/\* 不斷Loop直到都呼叫完畢,所以這部分父程序及子程序的步驟及迴圈次數如下 \*/

* + - 父： TELL\_CHILD(pid[n]) → WAIT\_CHILD() 【5\*52\*7】
    - 子： WAIT\_PARENT() → accumulation(d\_sum) → TELL\_PARENT(getppid())【52\*7】

1. 最後程式會把5年全部的電力消耗總和儲存在sum.txt,

請抓取總和除上所有天數 → 得到平均每日耗電量(Day\_Average) = 4743

