

CSIS0230A Principle of Operating Systems (Class A)

Tutorial 2

A multi-process program

Imagine that you have had a private mail that is so private that you encrypted it. But somehow you forgot some characters of the password that would decrypt the file. You did remember that the mail starts with the string "Dearest ". So you want to try all the possible passwords to see whether one would decrypt right. The following program is written.

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <crypt.h>
#include <string.h>

#define ENCRYPTED_FILE "enMail.txt"

/* Turn a block of bits returned by encrypt to 8 characters */
void block2chars(char block[64], char out[8]) {
    int i, j;
    for (i=0; i<8; ++i) {
        char x = 0;
        for (j=0; j<8; ++j)
            x = (x << 1) | block[i*8+j];
        out[i] = x;
    }
}

/* Turn 8 characters to a block acceptable by encrypt */
void chars2block(char in[8], char block[64]) {
    int i, j;
    for (i=0; i<8; ++i) {
        unsigned char x = in[i];
        for (j=0; j<8; ++j) {
            block[i*8+j] = (x >> 7);
            x <<= 1;
        }
    }
}

/* Turn 8 7-bits characters to a block acceptable by setkey */
void chars2key(const char in[8], char block[64]) {
    int i;
    char passwd[8];
    for (i=0; i<8; ++i)
        passwd[i] = in[i] << 1;
    chars2block(passwd, block);
}

/* Checks whether passwd decrypt encrypted to a string
starting with expectedStr */
int rightPassword(char *expectedStr, char *passwd, char *encrypted) {
    char block[64], key[64], str2[8];
    chars2key(passwd, key);
```

```
    setkey(key);
    chars2block(encrypted, block);
    encrypt(block, 1);
    block2chars(block, str2);
    return strcmp(str2, expectedStr, 8)==0;
}

int main() {
    int infile;
    char *str="Dearest ", encrypted[8];
    char passwd[8] = "abcxy";

    /* Open the file, read first 8 characters */
    infile = open(ENCRYPTED_FILE, O_RDONLY);
    if (infile == -1) {
        fprintf(stderr, "%s: %s\n", strerror(errno), ENCRYPTED_FILE);
        return -1;
    }
    read(infile, encrypted, 8);
    close(infile);

    /* Try to decrypt for all characters */
    for (passwd[5]=' '; passwd[5]<='~'; ++passwd[5])
        for (passwd[6]=' '; passwd[6]<='~'; ++passwd[6])
            for (passwd[7]=' '; passwd[7]<='~'; ++passwd[7])
                if (rightPassword(str, passwd, encrypted)) {
                    printf("The password is %.8s\n", passwd);
                    return 0;
                }
    return 1;
}
```

Conceptually the program is simple, although a bit long. The first 3 functions (*block2chars()*, *chars2block()*, *chars2key()*) are routines used for decryption, while *rightPassword()* decrypts a message and checks whether it starts with a particular string *expectedStr*. The *main()* function repeatedly call *rightPassword()* hoping that one will decrypt the mail correctly.

The **for** loops within *main* is the most time consuming part of the program. To speed things up, you decide to use a machine with multiple CPUs. But the program does not benefit from multiple CPUs (why?). You must **modify it so that multiple processes share the workload**.

- Step 1 Modify the program to create one more process, such that the two processes try different range of passwords.
- Step 2 Make the parent process waits for the child process before exiting.
- Step 3 Modify your program so that when one process finds the password, it informs the other process. The other process should then stop without trying all passwords.

Hints:

- You will need these system calls: *fork*, *mmap*, *waitpid*.
- Link the “crypt” library when compiling, i.e., “gcc crack.c -lcrypt”.