

Betriebssysteme Übung 9

Aufgabe 1

```
#include <unistd.h> // Header-Dateien laden
#include <stdio.h>

int p[2]; // Variablen initialisierern
int v;

void process()
{
    int token, pid;

    pid = getpid(); // holen der Prozess-ID
    printf("Process %d \n", pid); // Ausgabe der PID des Kindprozesses als Ganzzahl (%d)
    while(1) // Start einer Endlosschleife
    {
        read(p[0], &token, sizeof(int)); // p[0]=Filedeskriptor(lesen) - Wert der Größe int lesen und unter token
        // abspeichern
        printf("Process %d \n", pid); // Ausgabe der PID des aktuell laufenden Prozesses als Ganzzahl (%d)
        write(p[1], &token, sizeof(int)); // p[1]=Filedeskriptor(schreiben) - Wert der Größe int aus token lesen und
        // zum anderen Ende der Pipe schieben

        sleep(1); // schlafen für 1 Sek.
    }
}

int main()
{
    if(pipe(p) != 0) // prüfen ob Pipe angelegt wurde
    {
        perror("Error"); // wenn nicht: "Error" ausgeben
        return(1); // Rückgabewert von main = 1
    }

    for(v = 0; v < 2; v++) // for-Schleife wird 2x durchlaufen
    {
        if(fork() == 0) process(); // wenn für Sohnprozess Rückgabewert von fork gleich Null, dann Aufruf der
        // Funktion prosess
    }

    sleep(10); // schlafen für 10 Sek.
    write(p[1], &v, sizeof(int)); // p[1] Filedeskriptor(schreiben) - Wert der Größe int aus v an ein Ende der
    // Pipe schreiben
    wait(NULL); // auf Kindprozesse warten (keine Terminierung da endloses Arbeiten)
    return(0); // Rückgabewert von main = 0
}
```

In diesem Programm wird eine Pipe erzeugt. Anschließend wird durch einem Filedeskriptor p[1] in diese Pipe geschrieben und durch einen Filedeskriptor p[0] aus dieser Pipe gelesen.

Aufgabe 2

```
#include <stdio.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>

char choice[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789./";
// Feld mit gleichzeitiger Initialisierung

main(argc, argv) // Hauptprogramm bekommt argc (Anzahl der Argumente) und
// argv (Array mit eingegebenen Argumente) übergeben

int argc;
char **argv; // Zeigervariable (Zeigt auf Wert von argv)

{
struct timeval t; // t ist vom typ timeval (Zeittakt)
struct timezone tz; // tz ist vom typ timezone (Zeitzone)
int t1, t2, i;
char pwd[13], base[10], salt[3]; // Initialisierung von Arrays

if (argc < 2) // Stellt sicher das Programmaufruf mit mind. 1 Argument übergeben wird
{
printf(stderr, "Usage: crpwd \n"); // wenn nicht : Fehlermeldung auf Standard-Fehlerausgabegerät
exit(1); // Abbruch
}

gettimeofday(&t,&tz); // auslesen der Systemzeit
t1=(int)((float)(t.tv_usec-((t.tv_usec/100l)*100l))/1.5625);
t2=(int)((float)(t.tv_sec-((t.tv_sec/100l)*100l))/1.5625);
// Mit Hilfe der ausgelesenen Mikrosekunden und Sekunden werden
// 2 zufällige Werte erzeugt

salt[0]=choice[t1]; // Diese beiden Werte definieren die Nummer des Buchstaben (Index)
salt[1]=choice[t2]; // des oben vordefinierten Char-Array Choice, welche nun in das
salt[2]=0; // Char-Array salt übertragen werden . Wir erhalten eine Zeichenkette salt
// Beispiel : dF0

strcpy(base, argv[1]); // kopieren (strcpy) des 2. Argument (Beispiel: krypt Hallo) in base
strcpy(pwd, crypt(base, salt)); // base wird durch crypt mit Hilfe der Zeichenkette salt verschlüsselt und in
// pwd kopiert. Die ersten beiden Zeichen von salt bestimmen den
// Verschlüsselungsalgorithmus.

printf("%s\n", pwd); // Ausgabe von pwd auf Standardausgabegerät
}
```

Dieses Programm verschlüsselt also ein eingegebenes Argument (z.b. Passwort) mit Hilfe von zufälligen Werten, welche aus der aktuellen Systemzeit ermittelt werden, und gibt dieses Argument verschlüsselt auf dem Bildschirm aus.

Das Programm wird auf folgender Weise ausgeführt (Beispiel) : krypt Hallo

In diesem Falle stellt Hallo das Passwort dar (2. Argument)

Innerhalb des Programms wird sichergestellt, das auch tatsächlich ein 2. Argument eingegeben wurde.