



Betriebssysteme: Konzepte, Dienste,
Schnittstellen
(Betriebssysteme und betriebssystemnahe
Programmierung)

SS 2008 – Übungsblatt 10

Ausgabe: 2. Juli 2008

Abgabe: bis spätestens 9. Juli 2008

Aufgabe 1. *Hochfahren eines Linux-Systems*

Beschreiben Sie in eigenen Worten die verschiedenen Phasen beim Booten eines Linux/UNIX-Systems. Informieren Sie sich dazu bei

http://www.faqs.org/docs/linux_intro/sect_04_02.html

und

http://www.adminschoice.com/docs/booting_process_in_solaris.htm.

Aufgabe 2. *init*

Was ist die Aufgabe des Prozesses `init`?

```
root 1 0 0 Apr23 ? 00:00:36 init
```

Erläutern Sie den Inhalt von `/etc/inittab` und `/etc/init.d`.

Aufgabe 3. *inetd*

Welche Aufgaben erfüllt der Dämon `inetd`? Erläutern Sie den Inhalt der Datei `/etc/inetd.conf`.

Aufgabe 4. *Paßwort-Eingabe: ask_passwd2.c*

Bringen Sie das folgende Programm zum Ablauf

```
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <termios.h>

char *getpass(const char *);

#define MAX_PASS_LEN 8

int
main(void)
{
    char *ptr;

    if ( (ptr = getpass("Enter password:")) == NULL){
        fprintf(stderr, "%s\n", "getpass error");
        exit(1);
    }
    printf("password: %s\n", ptr);

    /* now use password (probably encrypt it) ... */

    while (*ptr != 0)
        *ptr++ = 0;    /* zero it out when we're done with it */

    exit(0);
}

char *
getpass(const char *prompt)
{
    static char buf[MAX_PASS_LEN + 1];
        /* null byte at end */

    char *ptr;
    sigset_t sig, sigsave;
    struct termios term, termsave;
    FILE *fp;
    int c;

    if ( (fp = fopen(ctermid(NULL), "r+")) == NULL)
        return(NULL);
    setbuf(fp, NULL);

    sigemptyset(&sig);
        /* block SIGINT & SIGTSTP, save signal mask */
```

```

sigaddset(&sig, SIGINT);
sigaddset(&sig, SIGTSTP);
sigprocmask(SIG_BLOCK, &sig, &sigsave);

tcgetattr(fileno(fp), &termsave);          /* save tty state */
term = termsave;                          /* structure copy */
term.c_lflag &= ~(ECHO | ECHOE | ECHOK | ECHONL);
tcsetattr(fileno(fp), TCSAFLUSH, &term);

fputs(prompt, fp);

ptr = buf;
while ( (c = getc(fp)) != EOF && c != '\n') {
    if (ptr < &buf[MAX_PASS_LEN])
        *ptr++ = c;
}
*ptr = 0;                                  /* null terminate */
putc('\n', fp);                            /* we echo a newline */

                                           /* restore tty state */
tcsetattr(fileno(fp), TCSAFLUSH, &termsave);

                                           /* restore signal mask */
sigprocmask(SIG_SETMASK, &sigsave, NULL);
fclose(fp);                                /* done with /dev/tty */

return(buf);
}

```

und erklären Sie seine Wirkungsweise Zeile für Zeile.

Aufgabe 5. *shell2*

Bringen Sie das folgende Programm zum Ablauf

```

#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXLINE 4096                                /* max line length */

int
main(void)
{

```

```

char    buf[MAXLINE];
pid_t   pid;
int     status;

printf("Input command: %% ");
while (fgets(buf, MAXLINE, stdin) != NULL) {
    buf[strlen(buf) - 1] = 0; /* replace newline with null */

    if ( (pid = fork()) < 0){
        perror("fork error");
        exit(1);
    }

    else if (pid == 0) {          /* child */
        execlp(buf, buf, (char *) 0);
        fprintf(stderr, "couldn't execute: %s ", buf);
        perror("execlp error");
        exit(127);
    }

    /* parent */
    if ( (pid = waitpid(pid, &status, 0)) < 0){
        perror("waitpid error");
        exit(1);
    }
    printf("%% ");
}
exit(0);
}

```

und erklären Sie seine Wirkungsweise Zeile für Zeile.