UMEÅ UNIVERSITET
Institutionen för datavetenskap
Gunnar Gunnarsson, Tomas Halvarsson

Tentamen Datavetenskap

Kurs: Systemprogrammering, 5p
Tentamensdatum: 21 november 1998 kl 9-15
Hjälpmedel: *A book on C* / *Vägen till C*
Examinator: Gunnar Gunnarsson (gunnar@cs.umu.se)

Personnummer: _____

Namn: _____

Namnteckning: _____

Maxpoäng: 40 poäng
Betygsgränser: 20 poäng (betyg 3), 26 poäng (betyg 4), 32 poäng (betyg 5)

Kryssa för de uppgifter som du har löst.

Uppgift	1	2	3	4	5	6	7	8	9	10	Totalt	Bonus
Löst												
Maxpoäng	4	4	4	6	2	3	4	3	4	6	40	2 (4)
Poäng												

Betyg: _____ Signatur: _____ Inlämningstid: _____

Kom ihåg att även ofullständigt lösta uppgifter kan ge poäng. Börja varje uppgift på ett nytt blad. Skriv endast på framsidan. Skriv ditt namn och uppgiftens nummer på varje blad. Observera att uppgifterna inte är ordnade i stigande svårighetsgrad. Tre sidor med funktionsprototyper finns bifogade. Använd också boken om du har den med dig, den innehåller i vissa fall mer information.

Uttryck dig **mycket tydligt**. Alla **otydligheter** tolkas till din **nackdel**.

Uppgift 1 (4 p)

Förklara kortfattat och mycket tydligt innebörden av följande termer i Unix-sammanhang:

a) dynamisk minnesallokering
b) blockerad signal
c) delat minne (shared memory)
d) miljövariabel
e) ickeblockerande I/O
f) fildeskriptortabell
g) semafor
h) minnesmappad I/O

Uppgift 2 (1+1+1+1=4 p)

a) Vad skrivs ut av nedanstående sats?

```
printf("%d\n", 1 ? (2 > 3) : (4 < 5));
```

b) Vilket värde kommer heltalsvariabeln i att ha när denna for-slinga avslutats?

```
for (i=0; i<23; i++);
```

c) Hur ser funktionsprototypen ut för en C-funktion vid namn F som tar ett argument bestående av en array med fem element av typen int, och returnerar en pekare till void?

d) Beskriv med ord vilken typ X har i följande deklaration (det första tecknet inom hakparenteserna är en nolla):

```
float (X)[0XA];
```

Uppgift 3 (3+1=4 p)

a) Förklara i grova drag hur make skulle behandla följande make-fil:

```
CC=gcc
CFLAGS=-Wall

lfdia:   lfdia.o
         gcc $(CFLAGS) -o lfdia lfdia.o -lm gd/libgd.a
         cp lfdia /Home/staff/gunnar/public_html/lf.cgi

lfdia.o: lfdia.c lfdia.h
         gcc $(CFLAGS) -c lfdia.c
```

Du bör bland annat beskriva vilka beslut programmet make fattar (och på vilka grunder), samt vilka kommandon som kommer att exekveras av make. Anta att filerna lfdia.c och lfdia.h existerar, medan lfdia och lfdia.o inte existerar. Du kan också anta att make inte får några kommandoradsargument.

b) Vad är vitsen med make-filer, bortsett från att man sparar skrivarbete?

Uppgift 4 (2+2+1+1=6 p)

```
#include <stdio.h>
#include <stdlib.h>

char *dup_str(const char *s2)
{
  char *s1, *tmp;

  s1 = malloc(strlen(s2+1));
  if ((tmp=s1) == NULL)
    return s1;

  while (*s1++ = *s2++);

  return s1;
}
```

Funktionen `dup_str()` ska ta en sträng och returnera en pekare till en kopia av strängen i ett nytt minnesområde, precis som biblioteksfunktionen `strdup()`. Om det inte går att allokera minne ska `dup_str()` returnera NULL.

Exempel: `dup_str("Sune")` ska returnera en pekare till en kopia av strängen `"Sune"`.

Koden kommer tyvärr inte att fungera som avsett, på grund av att den innehåller två mycket allvarliga fel.

a) Vilka är felen, och hur borde det vara istället?

b) Hur tror du att felen kommer att yttra sig när funktionen körs på de Sun Solaris-system som använts på laborationerna? Beskriv felen vart och ett för sig, dvs när du förklarar hur det ena felet yttrar sig ska du förutsätta att det andra felet är åtgärdat. Glöm inte att motivera dina teorier ordentligt.

c) `dup_str()` returnerar NULL om det inte gick att allokera minne. Man skulle också kunna tänka sig att skriva ut ett felmeddelande på stderr och avsluta programmet med ett anrop till `exit()`. Nämn ett bra argument för att man ska returnera NULL istället för att avsluta programmet med ett felmeddelande.

c) Nämn ett bra argument för att man ska avsluta programmet med ett felmeddelande istället för att att returnera NULL.

Uppgift 5 (1+1=2 p)

Som du kanske minns från föreläsningarna använder sig Unix-system av något som kallas set-user-ID-program (SUID). Ett sådant program har (lite förenklat) den egenskapen att när någon startar programmet körs den nya processen med samma rättigheter som den användare som äger programfilen.

a) Nämn ett Unix-program som använder SUID-teknik.

b) Istället för SUID skulle man kunna tänka sig att införa följande systemanropsfunktion:

```
int get_access(int uid);
```

Den process (vilken process som helst) som anropade funktionen skulle få samma rättigheter som den användare som anges av parametern uid. Det finns en mycket viktig anledning till att den här tekniken inte används i praktiken. Vilken är anledningen?

Uppgift 6 (2+1=3 p)

a) Ge exempel på två signaler som kan hanteras, och vad hanterare för dessa två signaler bör göra.

b) Ge kortfattade förklaringar av begreppen "hängande signal" och "leverans av signal".

Uppgift 7 (3+1=4 p)

a) Beskriv i 10-15 kortfattade punkter vad nedanstående program gör. Vad kallas denna sorts program?

```
#include <errno.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAXLINELEN 1025

int main(int argc, char *argv[])
{
    FILE *fp, *fd;
    char line[MAXLINELEN], in[MAXLINELEN];
    if (argc != 2)
        exit(1);
    if ((fp = fopen(argv[1], "r")) == NULL)
        exit(2);
    if (fgets(line, MAXLINELEN, fp) == NULL) {
        if (feof(fp))
            exit(0);
        exit(3);
    }
    while (!feof(fp) && (fgets(line, MAXLINELEN, fp) != NULL)) {
        if (strlen(line) == 0)
            continue;
        if (line[strlen(line)-1] == '\n')
            line[strlen(line)-1] = '\0';
        if (! strcmp("read", line)) {
            fgets(in, MAXLINELEN, stdin);
        } else if (! strncmp("exec ", line, 4)) {
            if ((fd = popen(line+4, "w")) == NULL) {
                perror(line+4);
                continue;
            }
            fprintf(fd, in);
            pclose(fd);
        } else {
            fprintf(stderr, "Error: \"%s\" not " \
                    "recognized.\n", line);
        }
    }
    fclose(fp);
    return 0;
}
```

b) Visa hur programmet kan ändras så att det ignorerar rader som börjar med "#". Du behöver inte skriva om allt, utan bara skriva kod för ändringarna.

Uppgift 8 (3 p)

Hur ser nedanstående kod ut efter att ha körts genom en C-preprocessor? Du kan anta att preprocessorn inte får några argument, och att detta är den enda kod som preprocessorn behandlar.

```
/* This section contains a percent function */
#ifndef __T_FKN__
#define __T_FKN__

#define MAXNUM 100
#define MAX (((a) > (b)) ? (a) : (b))

/* My percent function */
int percent(int val1, int val2)
{
    if ((max = MAX(MAX(val1, MAXNUM), val2)) == MAXNUM)
        return MAXNUM;
    else
        return max/MAXNUM;
}

#else

/* External declaration */
extern int percent(int val1, int val2);

#endif
```

Uppgift 9 (4 p)

Funktionen `popen` startar upp ett kommando, t.ex. ”wc –l”, och omdirigerar antingen stdin eller stdout för kommandot så att den process som anropar `popen` kan antingen skriva till kommandots stdin, eller läsa från kommandots stdout.

Skriv en egen version av funktionen `popen`, kallad `my_popen`. Funktionen ska ha följande funktionshuvud:

```
int my_popen(const char *command, char mode)
```

och således returnera en fildeskriptor, samt ta som argument dels en sträng innehållande det kommando som skall startas, och dels en char som anger om man vill skriva till kommandots stdin (w) eller läsa från kommandots stdout (r). Du behöver inte bekymra dig om signalhantering.

Uppgift 10 (1+2+3=6 p)

a) Hur många element innehåller arrayen `a` efter följande deklaration, och vad skrivs ut av printf-anropet?

```
char a[20] = "Tom sträng.\n";
printf("%s", a);
```

b) Kan man ändra värdet på `p` respektive `q` så att de refererar andra minnesceller, och isåfall hur? Kan man ändra innehållet i strängarna som `p` respektive `q` refererar till, och isåfall hur?

```
char q[] = "En sträng";
char *p = "En sträng";
```

c) För varje rad i programmet som påverkar `b[]`, visa hur innehållet i `b[]` ser ut efter det att raden exekverats. Du behöver inte förklara rader som inte påverkar `b[]`. Vad skrivs ut av programmet?

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char *s[] = {
        "Clownen Manne",
        "vill förorda och",
        "stipulera sträng",
        "tentarättning."
    }, *r, **p, b[40] = "";
    p = s;
    r = b;

    strcpy(b, (*p)+8);
    strcpy(r+3, &p[2][2]);
    r = &s[1][12];
    strcat(&b[strlen(b)-1], r);
    strncat(b, r-8, 4);
    b[strlen(b)] = *(r-8);
    b[strlen(b)-4] = *(*(p+3)+12);
    strncpy(&b[26], &s[3][5], 4);
    strcpy(&b[strlen(b)], *(p+3)+13);

    printf("%s\n", b);
    return 0;
}
```