Operativsystem ID1200/06

2020 01 07

Instruktioner

- Du får, förutom skrivmateriel, endast ha med dig en egenhändigt handskriven A4 med anteckningar. Anteckningarna lämnas in och kan inte återanvändas.
- Svaren skall lämnas på dessa sidor, använd det utrymme som finns under eller brevid varje uppgift för att skriva ner ditt svar.
- Svar skall skrivas på svenska eller engelska.
- Du skall lämna in hela denna tentamen och den handskrivna sidan med anteckningar. Inga ytterligare sidor skall lämnas in.

Betyg

Tentamen delas in i två delar, en grundläggande del bestående av fem frågor och en del för högre betyg också den bestående av fem frågor.

För godkänt, betyg E, skall den grundläggand delen klaras enligt nedan. För högre betyg krävs desutom att två eller flera av de fem övriga frågorna klaras med godkänt resultat.

- Fx: 7 poäng av 10 i den grundläggande delen
- E: 8 poäng av 10 i den grundläggande delen

Högre betyg ges basert på de övriga delarna.

- D: två godkänd
- C: tre godkända
- B: fyra godkända
- A: all godkända

Namn:_____ Persnr:_____

Svar: De angivna svaren är inte nödvändigtvis svar som skulle ge full poäng, längre förklaringar kan vara nödvändigt.

1 byta context [2 poäng]

Ett intressant experiment är att spara det nuvarande tillståndet för att sedan återinstallera det. Detta kan göras med systemanropen `getcontext()` och `swapcontext()`.

`The function getcontext(ucontext_t *ucp) initializes the structure pointed at by ucp to the currently active context.`

`The swapcontext(ucontext_t *oucp, const ucontext_t *ucp) function saves the current context in the structure pointed to by oucp, and then activates the context pointed to by ucp.`

Vad blir utskriften om man kör programmet nedan?

```
volatile int done = 0;

int main() {

  ucontext_t one;
  ucontext_t two;

  getcontext(&one);

  printf("done = %d\n", done);

  if(!done) {
    done = 1;
    printf(" - gurka -\n");
    swapcontext(&two, &one);
    printf(" - tomat -\n");
  } else {
    printf(" - salad -\n");
    swapcontext(&one, &two);
    printf(" - morot -\n");
  }

  printf(" - potät -\n");

  return 0;
}
```

Namn:_____ Persnr:_____

Svar:

```
done = 0
 - gurka -
done = 1
 - salad -
 - tomat -
 - potät -
```

Namn:_____ Persnr:_____

2 fork() [2 poäng]

Nedan finns lite skelett till kod som gör en `fork()`. Programmet fortsätter i två processer men skall göra olika saker i dessa. Komplettera koden så att de två sektionerna körs av de olika processerna och att den avslutande koden endast körs då barnet har terminerat.

```
int main() {

  // things we both do

  int pid = fork();

      // executed by child process
      :
      :
      return 0;


      // executed by the parent process
      :
      :


  // executed only when the child has terminated
  :
  :

  return 0;
}
```

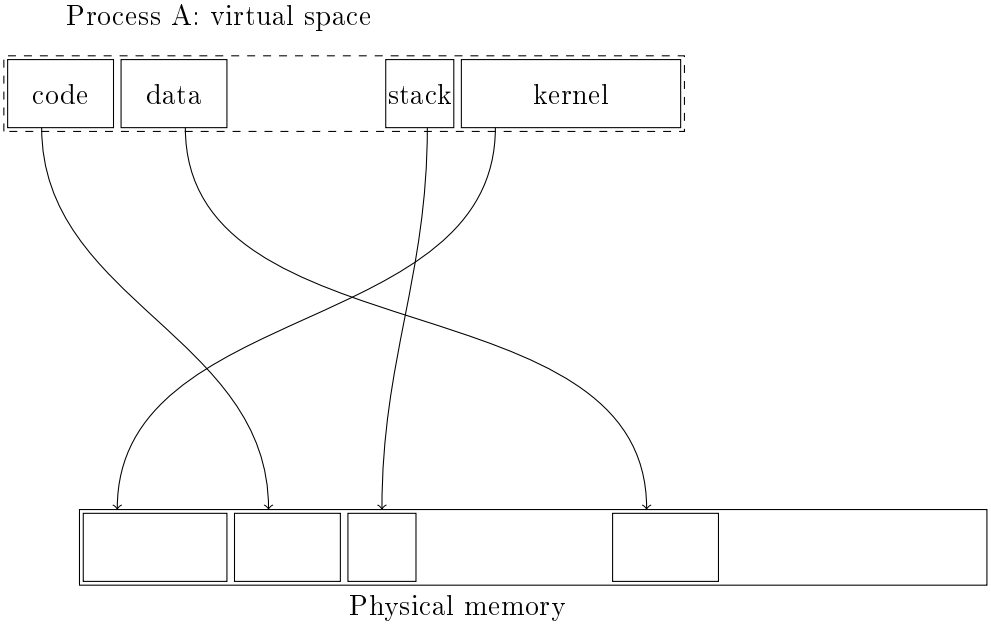
Svar:

```
int main() {
  :
  if(pid == 0) {
    // child
    return 0;
  } else {
    // mother
  }
  wait(NULL);
  // after
}
```

Namn:_____ Persnr:_____

3 Trådar och minne [2 poäng]

Nedan finns en bild av en process virtuella minne och hur dess segment är placerade i det fysiska minnet. Om processen skapar en ny tråd med ett anrop till `pthread_create()` så kommer bilden av minnet att förändras. Rita upp hur det virtuella och fysiska minnet har förändrats efter det att en ny tråd skapats.



Svar: En ny stack läggs in i det virtuella minnet och denna mappas till valfritt fritt segment i det fysiska minnet.

Namn:_____ Persnr:_____

4 a stack, a bottle and ... [2 poäng]

Du har skrivit programmet nedan för att undersöka vad som ligger på stacken med utskriften som följer.

```
void zot(unsigned long *stop ) {
  unsigned long r = 0x3;
  unsigned long *i;
  for(i = &r;  i <= stop; i++){ printf("%p          0x%lx\n", i, *i);  }
}

void foo(unsigned long *stop ) {
  unsigned long q = 0x2;
  zot(stop);
}

int main() {
  unsigned long p = 0x1;
  foo(&p);
 back:
  printf("  p: %p \n", &p);
  printf("  back: %p \n", &&back);
  return 0;
}
```

```
0x7fff2f14cf58          0x3
0x7fff2f14cf60          0x7fff2f14cf60
0x7fff2f14cf68          0xf3331713173eab00
0x7fff2f14cf70          0x7fff2f14cfa0
0x7fff2f14cf78          0x560ced18173c
0x7fff2f14cf80          0x1
0x7fff2f14cf88          0x7fff2f14cfb0
0x7fff2f14cf90          0x7f0d7750b9a0
0x7fff2f14cf98          0x2
0x7fff2f14cfa0          0x7fff2f14cfc0
0x7fff2f14cfa8          0x560ced18176a
0x7fff2f14cfb0          0x1
  p: 0x7fff2f14cfb0
  back: 0x560ced18176a
```

Rita ut var `zot:s` stackbas är och beskriv vad som ligger på adressen som vi får om vi adderar 24 bytes (hex 18) till den basen.

Svar: Basen ligger på `0x7fff2f14cf70` och 24 bytes ovanför hittar vi `0x7fff2f14cfb0` som är adressen på den lokala variablen `p` i `main()`.

Namn:_____ Persnr:_____

5 en flertrådad buffer [2 poäng]

Nedan finns koden för att plocka bort ett värde ur en buffer som kan anropas av flera trådar. Implementera motsvarande procedur `set(int i)` som lägger in ett värde i bufferten.

```
volatile int buffer = 0;
volatile int empty = TRUE;

pthread_cond_t added, removed;

pthread_mutex_t global;

int get() {
  int val;
  pthread_mutex_lock(&global);
  while(empty) {
    pthread_cond_wait(&added, &global);
  }
  val = buffer;
  empty = TRUE;

  pthread_cond_signal(&removed);
  pthread_mutex_unlock(&global);

  return val;
}
```

Svar:

```
void set(int i) {
  pthread_mutex_lock(&global);
  while(!empty) {
    pthread_cond_wait(&removed, &global);
  }
  buffer = i;
  empty = FALSE;

  pthread_cond_signal(&added);
  pthread_mutex_unlock(&global);
}
```

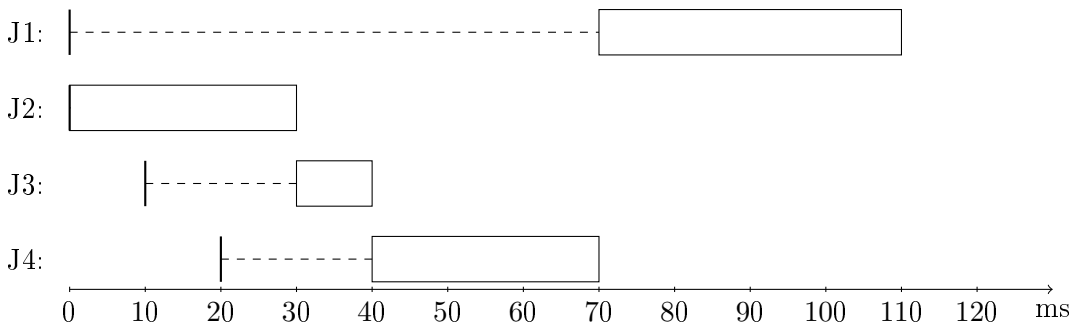
Namn:_____ Persnr:_____

6 Schemaläggning [P/F]

Antag att vi har en schemaläggare som använder *shortest job first*. Vi har fyra jobb som nedan anges med ⟨*anländer vid*, *exekveringtid*⟩ i ms. Rita upp ett tidsdiagram över exekveringen och ange omloppstiden för vart och ett av jobben.

Svar:

- J1 : ⟨0,40⟩ 110 ms
- J2 : ⟨0,30⟩ 30 ms
- J3 : ⟨10,10⟩ 30 ms
- J4 : ⟨20,30⟩ 50 ms



Namn:_____ Persnr:_____

7 sidindelat minne med sidor på 64 byte [P/F]

Du har som uppgift att föreslå en arkitektur för en processor som skall arbeta med att sidindelat minne där sidorna är så små som 64 byte. Processorn är en 16-bitars processor och den virtuella adressrymden skall vara 2^{16} byte.

Föreslå ett schema som använder sig av en hierarkisk sidtabell baserad på sidor om 64 byte och förklara hur adressöversättning går till.

Svar: Ett förslag är att använda ett offset på 6 bitar och sen ha två nivåer i trädet med index om 5 bitar för varje nivå. Sex bitar som offset är för att kunna adressera en sida på 64 byte. De 5 bitarna som index skulle ge 32 element i varje tabell och om vi har två byte per post så ger det en tabellstorlek på 64 byte. En post om två byte borde vara tillräckligt, vi kan till exempel använda 10 bitar för ramnummer och 6 bitar för diverse flaggor. Det skulle ge oss ett fysiskt minne på 64Ki byte.

Namn:_____ Persnr:_____

8 ett enkelt fs [P/F]

Antag att vi har ett enkelt filsystem uppbyggt med *inoder* och *datablock*. Rita upp hur vi kan representera en mapp och två filer, `foo.txt` på 5000 bytes och `bar.txt` på 200 bytes), med hjälp av dessa komponenter. Mappen skall ha länkar till de två filerna. Beskriv vilken information som finns var och vilka antagande du gör:

- filstorlek
- filägare
- länkar mellan namn och filer
- antalet länkar till fil
- skriv och läsrättigheter av fil
- innehåll av av fil

Svar:

Filerna representeras av inoder med pekare till de datablock som behövs. Om datablocken är 4 Ki byte stora så skulle `bar.txt` behöva ett block och `foo.txt` två block. Inoderna skulle inehålla: storlek, ägare,antalet nlänkar och rättigheter. Datablocken skulle naturligtvis ha filernas innehåll.

Mappen represnteras av en inode med en referens till ett datablock. I datablocket har vi en mappning mellan namnen på filerna (bar.txtöch foo.txt") och nummer på de inoder som används för filerna.

Namn:_____ Persnr:_____

9 malloc och free [P/F]

Antag att du tittar igenom en implementation av malloc där du vet att de fria blocken ligger i en enkellänkad frilista (`flist` i koden nedan). Blocken (`chunk`) har ett fält med sin storlek (`size`) och en pekare till nästa block (`next`). Det finns en funktion `after()` som givet ett block hittar det block som angränsar det givna blocket i minnet. Du kommer till koden för `insert()` nedan som skall lägga in ett block i frilistan men vad gör den mera?

Förklara vad koden gör och peka ut relevanta kodsegment som gör det du beskriver.

```
void insert(chunk *chnk) {
  chunk *aftr = after(chnk);
  chunk *next = flist;
  chunk *prev = NULL;
  while(next != NULL) {
    if( chnk == after(next) ) {
      next->size = next->size + sizeof(chunk) + chnk->size;
      if(prev != NULL) {
        prev->next = next->next;
      } else {
        flist = next->next;
      }
      chnk = next;
    }
    if(next == aftr) {
      chnk->size = chnk->size + sizeof(chunk) + next->size;
      if(prev != NULL) {
        prev->next = next->next;
      } else {
        flist = next->next;
      }
      next = next->next;
    } else {
      prev = next;
      next = next->next;
    }
  }
  chnk->next = flist;
  flist = chnk;
  return;
}
```

Namn:_____ Persnr:_____

Svar: Proceduren söker igenom free-listan för att se om det finns fria block omedelbart före och/eller efter (i adressordning) det block som skall läggas in. Om så är fallet så slås blocken samman till ett större block.

Namn:_____ Persnr:_____

10 Hej från en kärnmodul [P/F]

Du har implementerat en liten kärnmodul och lagt in den med hjälp av `insmod`. Modulen är skriven så att man kan få ut information, i detta fall strängen "Hello, hello :-)\n" men hur skall man få ut den? Beskriv hur du skulle gå tillväga för att att kommunicera med den här kärnmodulen och få strängen i retur.

```
static int hello_open(struct inode *inode, struct  file *file);

static const struct file_operations hello_fops = {
  .owner = THIS_MODULE,
  .open = hello_open,
  .read = seq_read,
  .llseek = seq_lseek,
  .release = single_release,
};

static int hello_show(struct seq_file *m, void *v) {
  seq_printf(m, "Hello, hello :-)\n");
  return 0;
}

static int hello_open(struct inode *inode, struct file *file) {
  return single_open(file, hello_show, NULL);;
}

static int __init hello_init(void) {
  proc_create("hello", 0, NULL, &hello_fops);
  printk(KERN_INFO "Hello in control\n");
  return 0;
}

static void __exit hello_cleanup(void) {
  remove_proc_entry("hello", NULL);
  printk(KERN_INFO "I'll be back!\n");
}

module_init(hello_init);
module_exit(hello_cleanup);
```

Svar: Enklaste sättet vore nog att via en terminal skriva:

```
cat /proc/hello
```