KTH - Skolan för Teknik och Hälsa

TENTAMEN I OPERATIVSYSTEM
HI1025:TEN1

21 mars 2014

Dataingenjörsprogrammet, årskurs 1

Instruktioner: Tentamen innehåller 11 frågor med totalt 47 poäng. För lägsta godkända betyg (E) krävs ungefär 23 poäng. För att få komplettera (Fx) krävs ungefär 21 poäng.

Viktigt: När du svarar på en fråga ska det i allmänhet framgå varför du vet svaret på en fråga, det uppnås enklast genom att du sätter in termer och begrepp som du hanterar i deras sammanhang på ett korrekt sätt. I frågor som kräver mer text har jag understött denna mekanism genom att kräva att vissa termer och begrepp finns i ditt svar och att dessa hanteras korrekt i sina respektive sammanhang. Men det är också viktigt att inte bli för mångordig, i uppgifter med 1 poäng krävs ett kort svar, i uppgifter med mer poäng behöver svaret utvecklas mer.

Anmärkning: Av layoutskäl saknas nödvändiga `#include`-direktiv i källkoden i vissa av uppgifterna, men uppgiften ska behandlas som om de fanns där.

Lycka till!

Johnny

1. Förklara hur så kallade *shared libraries* möjliggör att vi sparar primärminne när ett datorsystem kör med processer som använder sig av detta. **(3p)**

2. Då vi förlägger parallella aktiviteter i parallella processer skapar vi dessa parallella processer med `fork/exit/wait`. Och ibland en `exec`. Då vi förlägger parallella aktiviteter i parallella p-trådar, som hör till samma process, så skapar vi dessa parallella trådar med `pthread_create/pthread_exit/pthread_join`.

a) Jämför `wait` med `pthread_join`, vad finns för skillnader och likheter? *(2p)*
b) Vad händer om man i en process skapat många trådar och så anropar en av trådarna `exec`? *(1p)*
c) Ange och förklara en fördel med att förlägga parallella aktiviteter i parallella trådar istället för att skapa hela parallella processer för de parallella aktiviteterna. *(1p)*

(4p)

3. Vad skulle du säga om en person bad dig skapa en katalog på swappartitionen? **(2p)**

4. Studera nedanstående program.

```
main()
{
  pid_t pid;

  pid = fork();
  if(pid==0)exit(0);

  pid = fork();
  if(pid==0) {

    pid = fork();
    if(pid==0)
      exit(0);
    else
    {
      wait(0);
    }

    exit(0);
  }

  wait(0);
  wait(0);
}
```

a) Rita ett tidsdiagram som illustrerar släktförhållandena mellan de ingående processerna. Du behöver inte använda min stil att rita tidsdiagram, du kan ha en egen stil, det viktiga är tydligheten. Du kanske dessutom hittar ett bättre sätt än jag, då blir jag glad för jag vill att kursens innehåll ska bli bättre och bättre. **(3p)**

b) Hur många processer är maximalt igång samtidigt som ett resultat av detta programs körning? Motivera ditt svar genom att hänvisa till tidsdiagrammet du ritat. **(1p)**

5. Förklara följande termer.

a) Dynamisk laddning b) Filsystem c) Hård länk d) Semafor

(8p) *(2p styck)*

6. Vad menas med att trådar behöver *synkroniseras*? I din redogörelse ska en förklaring av begreppet *mutual exclusion* ingå. Ge ett illustrerande exempel. **(4p)**

7. Vad är en *signal* i ett *UNIX*-system och vad används de till? Ge exempel på två olika signaler. Vad menas med att fånga upp en signal? Det finns en signal som man inte kan fånga upp, vilken är det och varför går inte den att fånga? **(5p)**

8. Studera nedanstående två program: (`#include`-direktiv är utelämnade)

`a.c:`

```
main() {
  int to[2], from[2]; pid_t pid;
  pipe(to); pipe(from);
  char ch, message[] = "MESSAGE";

  pid=fork();
  if(!pid) {
    close(0); dup(to[0]);
    close(to[0]); close(to[1]);
    close(1); dup(from[1]);
    close(from[1]); close(from[0]);
    execlp("./b","./b",NULL);
  }

  ch=0;
  while(message[ch])
    write(to[1],&message[ch++],1);
  close(to[1]);

  wait(0);

  while(read(from[0],&ch,1))
    printf("%c",ch);
  printf("\n");

}
```

`b.c:`

```
main() {
  char ch;
  while(read(0,&ch,1))
  {
    ch+=32;
    write(1,&ch,1);
  }
}
```

`Asciitabell:`

32		33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~		

Vid behov kan ni få en större variant av denna tabell, be mig om detta på själva tentan så ordnar jag det.

Båda kompileras till de körbara programmen `a` respektive `b`. Dessa processer kommunicerar på något sätt. Hur? Man kan förstå hur det är tänkt att dessa processer ska kommunicera genom att studera omdirigeringarna i `a.c`. Beskriv hur denna kommunikation är tänkt att gå till och vad den ska resultera i för utskrift. Dock finns ett *problem* med `a.c`, det är något som *saknas*. Lägg till det som saknas och förklara vad anropet `./a` ger för resultat, förklara alla detaljer i hela programmets körning. Det ska också inkludera förklaringar av hur `b` kör (eftersom `b` också är inblandad). **(6p)**

9. Karakterisera de olika aktiviteterna i ett datorsystem som tillhörandes det som händer i operativsystemets kärna, skal/systemprogramvara och det som händer i användarapplikationer genom att sätta kryss i rätt ruta. Ett rättplacerat kryss ger 1 poäng. Ett felplacerat kryss ger –1 poäng, det är alltså riskabelt att chansa. (Att avstå ger 0 poäng.) Vi antar att vi har en liten kärna och vi anser att drivrutiner inte hör till kärnan utan är skal/systemprogramvara.

	Kärnan	Skal/systemprogramvara	Applikation	Avstår
Göra rättstavning i ett dokument i ordbehandlaren				
Skapandet av nya processer				
Skapandet av filer i ett filsystem				

(3p) *(Man kan inte få negativa totalpoäng på denna uppgift, lägst 0.)*

10. En server har följande principiella centrala kodavsnitt där den tar emot nya klientanslutningar och betjänar dem i funktionen `serve_client()`:

```
 while(1) {  // main accept() loop
    new_fd = accept(sockfd, ... );
    if (!fork()) { // this is the child process
        close(sockfd); // child doesn't need the listener
        serve_client(new_fd);
        close(new_fd);
        exit(0);
    }
    close(new_fd);  // parent doesn't need this
}
```

Hur kan man göra för att skriva om denna kod så att servern betjänar klienterna i parallella trådar istället för parallella processer? Vi kan säga att servern samtidigt max ska kunna hantera 5 klienter. Vilka ändringar måste man göra beträffande variabeln `new_fd` som innehåller en fildeskriptor … hur ska det här med fildeskriptorer hanteras då vi har trådar alltså? Det är rimligt att ha en övre gräns på antalet klienter som servern kan betjäna, det är då viktigt att denna övre gräns (här 5 stycken) finns med i dina diskussioner kring hur servern ska utvecklas. **(5p)**

11. En listning av filer i en *UNIX*-miljö har följande utseende

```
-rwxrw-rw- 1 johnny users    0 Mar 10 14:35 fil
drwxr-xr-- 2 johnny users 4096 Mar 10 14:35 katalog
```

Beskriv för dessa två filer rättigheter för ägaren, `johnny`, rättigheterna för gruppen `users`, och alla andras. **(3p)**