CONCEPTES AVANÇATS DE SISTEMES OPERATIUS (CASO)
Facultat d'Informàtica de Barcelona,
Dept. d'Arquitectura de Computadors,
curs 2018/2019 – 2Q
2n Control Parcial 26 d'abril de 2019

- L'examen és individual.
- Responeu cada pregunta en un full separat.
- Indiqueu a cada full COGNOMS, NOM i DNI.
- L'examen és sense llibres ni apunts, no es poden consultar ordinadors portàtils o mòbils.
- Justifiqueu totes les respostes.
- Temps d'examen: **90 minuts**.

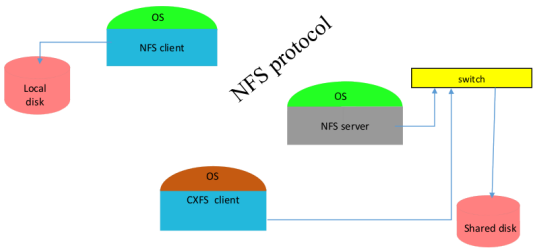
1. (1,5 punts) Sabent que parlem d'un sistema Mach, relaciona l'abstracció amb la seva definició:

(1) Thread	**(a)** is an execution environment that provides the basic unit of resource allocation.
(2) Memory Object	**(b)** is a source of memory. Tasks can access it by mapping portions of an object (or the entire object) into their address spaces.
(3) Port	**(c)** is the basic unit of execution and must run in the context of a task (which provides the address space).
(4) Task	**(d)** is the basic object-reference mechanism in Mach and is implemented as a kernel-protected communication channel.
(5) Message	**(e)** is a group of ports sharing a common message queue. A thread can receive messages for a port set and thus service multiple ports.
(6) Port set	**(f)** is the basic method of communication between threads in Mach. It is a typed collection of data objects.

1c, 2b, 3d, 4a, 5f, 6e.

2. (2 punts) Fent servir el següent dibuix, explica, raonant pros i contres, les diferències entre un Network Attached Storage (NAS) i una Storage Area Network (SAN). Fes especial esment en el sistema de fitxers.

NAS i SAN són dos enfocaments d'emmagatzament en xarxa que han demostrat la seva validesa. En tots dos casos, els usuaris accedeixen als seus fitxers de manera transparent, sense ser conscients de que d'altres usuaris poden estar accedint als mateixos fitxers concurrentment.



NAS depèn del protocol de Network File System(NFS), està, per tant, centralitzat en un servidor. S'implementa sobre Remote Procedure Calls (RPC). Tal com mostra el dibuix, un host servidor crea i exporta el sistema de fitxers i els hosts clients monten el sistema de fitxers exportat a sobre del seu, de manera remota. En resum, un NAS ofereix funcionalitats de sistema de fitxers (open, read, write, close) a clients remots.

A diferència del NAS, una SAN ofereix operacions més simples, de mida fixa, a nivell de bloc. És a dir, un host veu la SAN connectada a ell com si es tractés d'un disc tradicional. La tecnologia de connexió acostuma a ser Fiber Channel, amb un switch al que es connecten els hosts i l'array de discos. La coordinació d'accés als discos pertany a una capa superior, és a dir, per sobre de la SAN ha d'haver-hi un sistema de fitxers que gestioni els accessos a la SAN de tots els hosts. Un exemple d'aquest sistema de fitxers és CXFS, propietat de Silicon Graphics, que permet als clients (amb sistemes operatius heterogenis) connectats via fiber al switch accedir simultàniament als discos. Necessita d'un host (Linux) que actuï de *metadada broker*. Aquest, gestiona els locks entre els diferents accessos via TCP/IP i Ethernet.

Tots dos sistemes no són pas incompatibles. Podem tenir un array de discos en SAN, a on un dels clients locals fa de servidor NAS, tot exportant a sistemes remots el sistema de fitxers via NFS.

3. (1,5 punts) Explica la diferència entre la implementació habitual d'un pthread_mutex i els futex de Linux.

Futexes ("Fast user-space mutexes") són els maons més bàsics de Linux per construir abstraccions de bloqueig de nivell superior tals com els mutex de POSIX, les variables de condició, els semàfors, els *barriers* i els *read-write locks*.

També s'ha acceptat com a resposta vàlida (tot i que no està actualitzada), la solució proposada a l'examen de l'any 2013:

- pthread_mutex: sempre crida al sistema operatiu, amb un alt cost.
- futex: en els casos de contenció baixa, quan només un flux està accedint i sortint de la regió crítica, els futex permeten que l'accés i la sortida es facin

completament a nivell usuari. Només quan hi intervenen més fluxos cal cridar al sistema per bloquejar-se.

4. (2 punts) Defineix les següents mètriques d'avaluació de rendiment:

   a) Temps d'execució: és la diferència entre l'hora en què finalitza la tasca i l'hora en què va començar a córrer la tasca. S'acostuma a diferenciar entre *elapsed time* que és el temps total d'execució i *cpu time* que és el temps emprat activament per la tasca fent servir els recursos del processador.

   b) Acceleració (*speedup*): la relació entre el temps d'execució en seqüencial i el temps d'execució en paral·lel.

   c) Ample de banda (*bandwitdh*): la relació entre les dades transmeses i el temps que s'ha invertit en transmetre-les.

   d) Latència: cost d'iniciar operacions o comunicacions.

5. (1 punt) Què és el "system consolidation"?

   Agrupar diverses màquines físiques en una sola màquina virtual. Es fan servir les màquines físiques per provar que un servei funciona i determinar quants recursos necessita, per llavors moure'l a una màquina virtual amb aquests recursos ajustats.

6. (2 punts) Donat aquest codi, respon raonament a les qüestions:

   1. És codi de kernel o de usuari?
   2. Quin tipus de dispositiu vol tractar?
   3. Com cridaries a la funció de llegir del dispositiu des d'usuari? I des de l'interior del kernel?
   4. Què representa, a nivell de sistema, la variable MY_MAYOR? Pots saber quant val?

```
static int mychardrv_init(void) {
...
res = register_chrdev(MY_MAJOR, "mydrv", &mydrv_fops);
if (res < 0) {
    printk("unable to get major %d for mydrv\n", MY_MAJOR);
return res;
}
...
}
```

1. És codi de kernel, només cal fixar-se en les funcions `printk()` i `register_chrdev()`.
2. En cridar a `register_chrdev()` està registrant un dispositiu de caràcter.
3. Des de codi d'usuari, cridant a la funció `read(fildes, buf, nbyte);`. Des de kernel, cridant a la funció `mydrv_fops.read(file, buf, count, pos );`
4. El major identifica la família de dispositius i, per tant, el driver associat al dispositiu. Pots assignar-li un valor d'inici o deixar que el kernel ho faci. En tot cas, en tornar de la crida a `register_chrdev()`, aquesta variable està inicialitzada i per tant pots saber quant val.