Segon Parcial CASO - Victor Correal Ramos

Pregunta 1

Un sistema operatiu és un programari que ha de controlar tot el hardware de la màquina i proporcionar una interfície adient per tal que l'usuari faci servir la màquina. Aquest sistema operatiu ha de garantitzar seguretat, eficiencia i robustesa sobre el hardware.

El sistema operatiu MSDOS del 1982 era un dels primers sistemes operatius i el fragment de codi ja presenta el tret de seguretat (hi ha control d'errors en aquesta entrada a sistema). Els altres dos objectius de la meva definició haurien de complir-se també:

- Un SO sense robustesa no tindria gaire utilitat, si l'usuari perd les seves dades deixara d'utilitzar aquest sistema.
- L'eficiència serà relativa a l'época, segurament l'estructura software ha evolucionat desde 1982 (per exemple 9 anys després amb el Linux) i també s'hauria afegit/millorat el suport hardware per determinades operacions tipiques dels sistemes operatius

Per mi, si compleix aquest criteris es un sistema operatiu.

Nou anys després el hardware va evolucionar i també la tecnología de fabricació, afectant en el temps d'execucció, però en l'arquitectura del software no haura variat molt (segurament l'entrada a sistema del fragment de codi sigui la mateixa). Es veuria benificiada l'eficiencia, doncs en el mateix temps la màquina Intel 80386 hauria de ser capaç de tenir més rendiment que la Intel 8088.

Cal destacar que a nivell de prestacions la Intel 8088 té 8 bits i la Intel 80386 en té 32, això donaria mes capacitat d'emmagatzemar dades al usuari. Personalment, les prestacions de cada màquina no afecten l'eficiència del sistema operatiu, ja que el codi de sistema depen directament de l'implementació de la màquina.

Pregunta 2

a)¿Quin entorn creus que representa, SAN o NAS?

> L'entorn representat és un SAN, ja que l'imatge presenta una xarxa dedicada al moviment de dades.

b) Explica breument, basant-te en el dibuix, com funciona.

> La figura mostra que les dades están a un servidor, i els altres equipaments representen:
>
> - Una consola per fer-hi tasques de manteniment i administració.(Storage Mangment Console). Presumiblement es poden aplicar, com a administrador del sistema, un sistema de quotes
> - Un sistema de cintes, que podria actuar com:
>   - Redundància de dades per poder fer recuperacions del Storage
>   - Journaling de les operacions contra l'storage
> - Dues xarxes a cada equip:
>   - Xarxa vermella, es la que accedeix directament al Storage
>   - Xarxa verda, es la xarxa que utilitza les cintes i l'Storage

c) Quina és la diferència fonamental entre SAN i NAS?

> Fonamentalment, NAS funciona nivell de fitxer i no requereix d'una xarxa dedicada.
>
> Pel contrari, SAN si que es monta sobre una xarxa dedicada i serveix dades a nivell de bloc.

d) Cita algun sistema de fitxers pensat explícitament per aquest entorn

>

Pregunta 3

Sembla que a alt nivell el programa esta creant fluxos d'execucció (reservant memoria amb mmap i clonant) i espera amb la crida futex a que el fill el desperti. Segurament el codi sigui fork(...) i després futex(...).

Després de la crida a futex, sembla que el mateix procès torna a repetir el fork(...) i futex(...).

Les crides a a sched_priority es podrien traduïr a al nivell sobre alguna crida de Pthreads.

Pregunta 4

mmap vs malloc

> Malloc es una crida mes específica que mmap, per aquest motiu utilitzaria malloc si només vull reservar memoria dinàmica per un array o similar.
>
> D'altra banda, utilitzara mmap si haig de reservar memoria per estructures que requereixin un control de lectura/escritura o una localització més especifica de memòria.

mmap vs read

> Read es una crida especialitzada en llegir un file descriptor de la taula de canal.
>
> D'altra banda, mmap posiciona a memoria un cert arxiu i però l'interficie es molt mes farragos (mes paràmetres).
>
> Personalment, jo utilitzaria read sempre.

Pregunta 5

a)

```
long copy_to_user (void __user * to, const void * from, unsigned long n){
    if(n<0) return PANIC_ERROR; //error intern del sistema
    while(n > 0){
        if(bad_addres(to)){
          errno = EFAULT;
          return n;
        }
        *to++ = *from++;
        n -= sizeof(void __user*);
    }
    return 0;       //OK
}
```

b)