Examen de Sistemas Operativos ITIS Fuenlabrada — Junio 2010

Tiempo total: 3 horas.

Problema: Llamadas al sistema (3,5 puntos)

Implemente para Plan 9 un programa llamado `catz.c` que concatene a su entrada estándar el fichero que se le pasa como único argumento para después comprimirlo ejecutando gzip. Se podría utilizar como sigue:

```
; catz fichentrada2 < fichentrada1 > fichsalida.gz
```

En esta ejecución, el comando crea un fichero comprimido `fichsalida.gz` que contiene los datos del fichero `fichentrada1` seguidos de los datos del fichero `fichentrada2`. Los datos se deben comprimir con un nivel de compresión 9, mediante la ejecución de:

```
gzip -9
```

No se permite usar ficheros temporales.

Entrega: Copie un fichero llamado `catz-`*login*`.c`, **donde** *login* **es tu nombre de usuario,** en el directorio `/usr/elf/examen`. **ATENCIÓN: Sólo se puede efectuar la entrega una vez.**

Problema: Script de Shell (3,5 puntos)

Cree un script en RC llamado `hasstderr.rc` que imprima por su salida estándar una lista con los procesos que tienen su **salida estándar de errores** redirigida al fichero especificado como primer argumento. Si no se pasa ningún argumento, el script debe indicar su forma de uso y acabar con el estatus apropiado. La lista debe indicar el PID del proceso y el nombre del programa que ejecuta. Por ejemplo:

```
; hasstderr.rc /dev/cons
1       init
11      fossil
113     plumber
```

Se puede considerar que el nombre del fichero (el argumento) no contiene ningún metacarácter usado en expresiones regulares. Recuerde de Plan 9 ofrece información sobre los procesos en el directorio `/proc`.

Entrega: Copie un fichero llamado `hasstderr-`*login*`.rc`, **donde** *login* **es tu nombre de usuario,** en el directorio `/usr/elf/examen`. **ATENCIÓN: Sólo se puede efectuar la entrega una vez.**

Problema: Teoría (3 puntos)

Responda en un folio de examen las siguientes cuestiones:

A) En un planificador Round-Robin ejecutando en un sistema en el que un cambio de contexto tarda 1 ms, ¿Qué valor de los siguientes te parece más apropiado para el cuanto?

```
(1) 2 ms
(2) 20 ms
(3) 2000 ms
```

Razone la respuesta.

B) ¿Qué es el *thrashing* si hablamos de memoria virtual? ¿Cómo se evita?

C) ¿Para qué sirve *copy-on-write* cuando hablamos de paginación y por qué se puede realizar?

Solución

catz.c

```
#include <u.h>
#include <libc.h>

static void
usage(void)
{
	fprint(2, "usage: %s fich...\n", argv0);
	exits("usage");
}

static int
copy(int dfd, int sfd)
{
	char	buf[1024];
	int	nr;

	for(;;){
		nr = read(sfd, buf, sizeof buf);
		if(nr <= 0)
			return nr;
		if(write(dfd, buf, nr) != nr)
			return -1;
	}
}

int
pipeto(char *file, char *cmd, char *arg)
{
	int	fd[2];

	if(pipe(fd) < 0)
		return -1;
	switch(fork()){
	case -1:
		close(fd[0]);
		close(fd[1]);
		return -1;
	case 0:
		close(fd[1]);
		dup(fd[0], 0);
		close(fd[0]);
		execl(file, cmd, arg, nil);
		sysfatal("exec: %s: %r", file);
	default:
		close(fd[0]);
	}
	return fd[1];
}

void
main(int argc, char *argv[])
{
	int	ofd;
	int	ifd;
```

```
        ARGBEGIN{
        default:
                usage();
        }ARGEND;
        if(argc < 1)
                usage();

        ofd = pipeto("/bin/gzip", "gzip", "-9");
        if(ofd < 0)
                sysfatal("pipeto: %r");
        if(copy(ofd, 0) < 0)
                sysfatal("copy /fd/0 failed: %r");
        for(; argc > 0; argc--, argv++){
                ifd = open(argv[0], OREAD);
                if(ifd < 0)
                        sysfatal("open: %s: %r", argv[0]);
                if(copy(ofd, ifd) < 0)
                        sysfatal("copy %s failed: %r", argv[0]);
                close(ifd);
        }
        close(ofd);
        exits(nil);
}
```

—

hasstderr.rc

```
#!/bin/rc

rfork e

if(! ~$#* 1){
        echo 'usage: hasstrerr.rc filename' >[1=2]
        exit usage
}

for(i in `{grep -l '^  2[ →]+.*'^$1^'$' /proc/*/fd}){
        p=`{echo $i | sed 's%/proc/(.*)/fd%\1%'}
        n=`{cat /proc/$p/status >[2] /dev/null | awk '{print $1}'}
        if(! ~ $#n 0)
                echo $p '→' $n
}

exit ''
```

—

TEORÍA

A) La política de planificación Round-Robin, si el cuanto es demasiado grande, degenera en una política FIFO, y como resultado las aplicaciones interactivas sufren especialmente. Un cuanto de 200 ms podría llevar a esa situación. Por otra parte, si el cuanto es muy pequeño, se corre el riesgo de malgastar demasiado tiempo de CPU en cambios de contexto. En el caso del cuanto de 2 ms, si un cambio de contexto tarda 1 ms, se estaría perdiendo aproximadamente 1/3 del tiempo de CPU. Por tanto, lo más razonable sería elegir el cuanto de 20 ms.

B) Es la situación en la que hay demasiados procesos en ejecución para la memoria física disponible en la máquina, estos se roban marcos de página continuamente entre sí (suponiendo una asignación

global) porque se desalojan las páginas que necesitan acceder frecuentemente, provocando expulsiones y recuperaciones del área de intercambio o *swap*. Esta situación hace que degenere gravemente al rendimiento de la máquina.

Se puede prevenir evitando la ejecución de procesos cuando se estima que hay riesgo de que los procesos dejen de tener marcos para sus páginas más usadas. Para ello, se estima el *conjunto de trabajo* de los procesos que se están ejecutando. El conjunto de trabajo es el conjunto de páginas que usa un proceso dentro de una ventana de tiempo. Por tanto, en base a la suma de todos los conjuntos de trabajo, el sistema estima si la creación de un nuevo proceso puede provocar riesgo de *trashing*.

C) En general, *copy-on-write* es una estrategia para evitar el mantenimiento de distintas copias del mismo recurso mientras que dichas copias sean indistinguibles entre sí. A nivel de páginas de memoria virtual, si dos procesos comienzan con una página con los mismos datos, pueden tener asociado el mismo marco de página mientras que sólo estén leyendo los datos, ya que no hay diferencia entre leer los datos del mismo marco y leer los datos de dos marcos con datos exactamente iguales. Una vez que uno de los procesos modifique su página, se tiene que asignar un marco propio asociado a su página, ya que a partir de ese momento los datos difieren.

—