CC4302 Sistemas Operativos – Control 3 – Semestre Primavera 2016 – Prof.: Luis Mateu

Pregunta 1

I. (3 puntos) La siguiente es la implementación de la estrategia del working set:

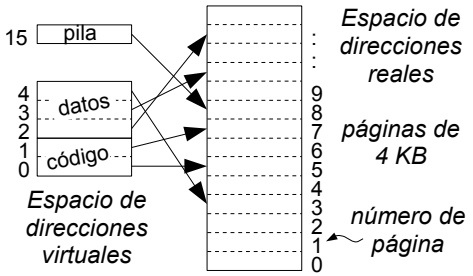
```
// Invocada para recalcular el working set
void computeWS(Process *p) {
  int *ptab= p->pageTable;
  for (int i= p->firstPage;
       i<p->lastPage; i++) {
    if (bitV(ptab[i]) {
      if (bitR(ptab[i])) {
        setBitWS(&ptab[i], 1);
        setBitR(&ptab[i], 0);
      }
      else {
        setBitWS(&ptab[i], 0);
} } } }

// Invocada cuando ocurre un page-fault
void pageFault(int page) {
  Process *p= current_process;
  int *ptab= p->pageTable;
  if (bitS(ptab[page]))
    send(ws, &page); // ws ejecuta
                     // WSStrategy
  else
    segfault(page);
}

// Un proceso daemon
void WSStrategy() {
  Process *p;
  int needsSwapping= 1;
  int page= *(int*)receive(&p);
  // proceso p gatilla el page-fault de page
  Iterator *it=processIterator();
  for (;;) {
    Process *q= nextProcess(it);
    int *qtab= q->pageTable;
    for (int i= q->firstPage;
         i<q->lastPage; i++) {
      if (bitV(qtab[i]) &&
          !bitWS(qtab[i]) &&
          !bitR(qtab[i])) {
        // se reemplaza pagina i del proceso q
        int real_page=realPage(qtab[i]);
        savePage(q, i);
        setBitV(&qtab[i], 0);
        setBitS(&qtab[i], 1);
        int *ptab= p->pageTable;
        setRealPage(&ptab[page],
                    real_page);
        setBitV(&ptab[page], 1);
        loadPage(p, page);
        setBitS(&ptab[page], 0);
        purgeTlb(); // invalida TLB
        purgeL1();  // invalida cache L1
        needsSwapping= 0;
        reply(p);
        page= *(int*)receive(&p);
    } }
    if (!hasNext(it)) {
      if (needsSwapping) {
        // No hay reemplazo posible
        doSwapping();
      }
      reset(it);
      needsSwapping= 1;
} } }
```

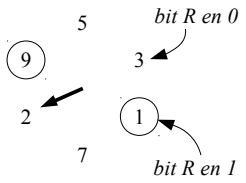
Reimplemente la misma estrategia pero considerando una MMU que *no* implementa el bit R (*reference*). Ayuda: use astutamente el bit de validez en su reemplazo, agregando otro bit que indique la validez efectiva de una página.

II. (1,5 puntos) La figura de la izquierda muestra la asignación de páginas virtuales de un proceso a páginas reales de un sistema Unix que implementa *fork* usando la técnica *copy on write*.



Suponga que el proceso llama a *fork*. Indique el contenido de las tablas de páginas del proceso padre y del proceso hijo justo después que el hijo escribe en la página virtual 15 y el proceso padre escribe en la página virtual 3. Incluya en las tablas: número de página virtual, número de página real y atributos de validez y escritura (V y W).

III. (1,5 puntos) Considere un sistema Unix que implementa la estrategia del reloj. El sistema posee 6 páginas reales disponibles y corre un solo proceso. La figura indica el estado inicial de la memoria, mostrando las páginas residentes en memoria, la posición del cursor y el valor del bit R.



Dibuje los estados por los que pasa la memoria para la siguiente traza de accesos a páginas virtuales: 9, 2, 6, 1, 2, 8.

Pregunta 2

a.- (3 puntos) El siguiente cuadro muestra a la izquierda el código que lee del dispositivo */dev/mem* visto en cátedra:

Implementación actual	***Ejemplo de uso***

`static ssize_t mem_read(`
`  struct file *filp, char *buf,`
`  size_t count, loff_t *f_pos) {`
`  down(&mutex);`
`  if (count > curr_size-*f_pos) {`
`  count= curr_size-*f_pos;`
`  }`
`  copy_to_user(buf,`
`  mem_buf+*f_pos, count);`
`  *f_pos+= count;`
`  up(&mutex);`
`  return count;`
`}`
	`$ echo tal > /dev/mem`
	`$ cat < /dev/mem`
	`l`
	`$ cat < /dev/mem`
	`a`
	`$ cat < /dev/mem`
	`t`
	`$ cat < /dev/mem`
	`$`

Modifique la función *mem_read* de manera que cada vez que se abra el archivo en modo lectura, se obtenga un solo caracter y en orden inverso a como fueron escritos. El cuadro de arriba muestra a la derecha un ejemplo de uso.

No necesitará modificar el resto de las funciones del driver. Observe que el comando *cat* primero abrirá el archivo. Luego usará *read* para leer muchos caracteres (*count* >1). Ud. debe entregarle a lo más un caracter. En seguida *cat* invocará *read* una 2^{da} vez para leer muchos caracteres nuevamente. Ud. deberá entregarle 0 caracteres para señalar el fin del archivo. Ud. puede darse cuenta que es la 2^{da} vez que *cat* invoca *read* porque **f_pos* será mayor que 0. Finalmente *cat* cerrará el archivo.

b.- (1,5 puntos) ¿Para qué sirven los números *major* y *minor* de un dispositivo?

c.- (1,5 puntos) Compare las dos estrategias de paginamiento en demanda vistas en el curso desde el punto de vista de (i) sobrecosto en tiempo de ejecución cuando la memoria física sobra, (ii) page-faults cuando hay penuria de memoria pero hay un solo proceso en ejecución, (iii) page-faults cuando hay penuria de memoria y hay muchos procesos en ejecución.