Esercizi su OS161 (tratti da compiti di esame) - SOLUZIONI

1. È possibile realizzare la `sys_waitpid` utilizzando per l'attesa un lock, su cui fare `lock_acquire`, mentre la segnalazione, da parte della `sys_exit` viene realizzata con `lock_release` dello stesso lock? (motivare la risposta).

No, un lock non può essere usato per trasmettere sincronizzazioni di tipo wait-signal, proprio per il problema dell'ownership: il lock serve per mutua esclusione. Per wait-signal si consigliano semafori o condition variable (l'eventuale lock associato serve per mutua esclusione, non per gestire wait-signal).

2. Si vogliono realizzare le system calls `sys_open` e `sys_close`. È necessario associare a un file descriptor il concetto di ownership da parte di un thead, in modo tale che solo il thread che ha fatto la open di un file sia autorizzato a chiamare la close del file? (motivare la risposta).

No, un file non ha concetti di ownership di questo tipo: un file può essere chiuso da un thread diverso da quello che lo ha aperto. Indirettamente invece, un file ha un concetto di ownership legato al processo, in quanto un file descriptor è associato al contesto di un processo (e della relativa tabella dei file aperti).

3. A cosa servono le funzioni OS161 `copyin` e `copyout`?

Servono a effettuare copie di dati tra memoria user e kernel, copyin ha come destinazione la memoria kernel, dualmente la copyout. La principale differenza rispetto ad altre forme di copia da memoria a memoria consiste nel fatto che vengono gestite in modo consistente eventuali errori/eccezioni legate a indirizzi/puntatori user non validi, impedendo così al kernel di terminare in modo anomalo (es. crash/panic).

4. Si riporta una parte della funzione `as_define_region` (file `dumbvm.c`). Si supponga di ricevere, per i parametri `as` e `vaddr` i valori (esadecimali) `0x80048720` e `0x412370`, e per `sz` il valore (decimale) `4128`. Ricordando che `PAGE_SIZE` è definito come `4096`, e `PAGE_FRAME` come `0xfffff000` si simulino le istruzioni proposte, indicando, per ognuna, in esadecimale, i valori degli operandi e del risultato (il valore assegnato alla variabile).

```
as_define_region(struct addrspace *as, vaddr_t vaddr, size_t sz, int
                 readable, int writeable, int executable) {
   size_t npages;
   /* Align the region. First, the base... */
   sz += vaddr & ~(vaddr_t)PAGE_FRAME;
   vaddr &= PAGE_FRAME;
   /* ...and now the length. */
   sz = (sz + PAGE_SIZE - 1) & PAGE_FRAME; npages = sz / PAGE_SIZE;
   ...
}
```

Si supponga che il valore di `sz` ricevuto sia inferiore alla dimensione di una pagina (es 4090). È possibile che si ottenga per `npages` il valore 2? (motivare la risposta)

sz += vaddr & not(PAGE_FRAME) => 0x00001020 + 0x00412370 & 0x00000FFF = **0x00001390**

vaddr &= PAGE_FRAME => 0x00412370 & 0xFFFFF000 = **0x00412000**

sz = (sz + PAGE_SIZE -1) & PAGE_FRAME => (0x00001390 + 0x00001000 - 1) & 0xFFFFF000 = 0x00002369 & 0xFFFFF000 = **0x00002000**

npages = sz / PAGE_SIZE => 0x2000 / 0x1000 = **2**

Sì, è possibile, in quanto il segmento viene allineato a un multiplo di pagina sia all'inizio che alla fine. In questo caso c'è una forma di frammentazione interna sia sulla prima che sull'ultima pagina. Con `4090 = 0xFFA`, `sz` avrebbe prima assunto il valore `0xFFA+0x370 = 0x126A` e, infine, `npages = 2`.