

Semafori

<code>sem_wait(sem);</code>	Esegue <code>counter--</code> e nel caso aggiunge a waiting il processo.
<code>sem_signal(sem);</code>	Rilascia un gettone. Esegue <code>counter++</code> ed estrae dalla lista dei waiting.
<code>sem_ini(\$gettoni);</code>	Restituisce il primo semaforo libero inserendovi \$gettoni gettoni. 0xFFFFFFFF se qualcosa è andato storto.
<code>sem_valid(id);</code>	Verifica la validità di un semaforo.

Mutex: \$gettone = 1
 Sync: \$gettone = 0
 Nota: tutti gli errori sono 0xFFFFFFFF
 Attenzione: Tutti gli id sono natl

Processi

<code>des_p(id);</code>	Restituisce un <code>des_proc*</code> a partire dall'id.
<code>c_abort_p();</code>	Porta all'interruzione del processo. Non effettua return ed è utile nei casi di errore.
<code>p-> con testo[I_RAX] = \$value;</code>	Ritorno della funzione di value.
<code>rimozione_lista(\$lista);</code>	Rimuove la testa dalla lista (max prio)
<code>inspronti();</code>	Inserisce esecuzione in testa a pronti

Tutti gli id sono natl

Preemption

```
des_proc* work = rimozione_lista($lista);
inspronti();
inserimento_lista(pronti, work);
schedulatore();
```

Alla fine ci sarà sicuramente uno tra il processo in esecuzione e work attivo

Utilità

`flog(MODE, "msg");` Mostra a schermo il messaggio msg

LOG_DEBUG: costante per debug
 LOG_INFO: costante per informazioni
 LOG_WARN: costante per avviso

Tab iter

```
tab_iter it(tab, begin, dim);

it.down();
```

`it.is_leaf();`

`tab_entry& e = it.get_e();`

`vaddr v = it.get_v();`

`paddr p = extr_INDIR_FISICO($e);`

`set_INDIR_FISICO($e, $paddr);`

```
alloca _tab());
```

All
fra
lib
de
a
co
un
di
tra
res
ue
l'in
del

```
copy_des( paddr src, paddr dst, natl i, natl n)  
;
```

Co
de
da
dst
pa
dal

```
set_des(paddr dst, natl i, natl n, tab_entry e)  
;
```

Se
val
su
de
a p
dal
'i'

Utilizzo tab_iter

```
for (tab_iter it(p->cr3, ini_uhn_p, dim); it;  
it.next()) {  
    if (!it.is_leaf())  
        continue;  
  
    tab_entry & e = it.get_e();  
    vaddr v = it.get_v();  
    paddr p = extr_INDIR_FISICO(e); // or set  
    /* e &= ~BIT_RW;  
       abbiamo modificato RW: invalidamo  
       le entrate nel TLB  
       invalida_entry_TLB(v); */  
}
```



By Gray00

cheatography.com/gray00/

Published 7th June, 2022.

Last updated 27th June, 2022.

Page 1 of 6.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Memoria Virtuale	Funtore
trasforma(vaddr);	Restituisce il paddr relativo a un vaddr
tipo_livello_accesso	Vedere le note
alloca_frame();	restituisce l'indirizzo del frame caricato
rilascia_frame(\$paddr);	Libera il frame in cui è contenuto \$paddr
paddr base = p & ~0xFFF;	base di un frame
natl indice = p / DIM_PAGINA;	Indice di
invalida_entrat_a_TLB(\$vaddr);	Elimina da lista
cr0	des_proc elimina_da_lista(des_proc& testa, des_proc* p) { del TLB del processo while (scorri && scorri != p) { in prec = &scorri->puntatore; esecuzione scorri = scorri->puntatore; Abilita la pagina (scorri) zione *prec = scorri->puntatore; Non return scorri; implem- entato Rimuove dalla lista puntata da \$testa il des_proc* p. Indirizzo che ha causato page fault
cr1	paddr
cr2	albero di traduzione del processo
cr3	array dei descrittori di frame
vdf[];	
memcpy((void) dst, (void) src, int nbytes);	copia da dst a src nbytes

```
map(p->cr3, vaddr begin, vaddr end, FLAGS, putpaddr)
;
```

Mappa gli indirizzi fisici passati da putpaddr nell'intervallo [begin, end) restituendo il primo che non è riuscito a tradurre

```
unmap( p->cr3, vaddr begin, vaddr end, getpaddr);
```

Rimuove le traduzioni nell'intervallo [begin, end) da p->cr3

- **tipo:** ini, fin

- **livello:** utn, sis

- **accesso:** p, c

Livello tabelle: 4, 3, 2, 1

FLAGS: bit_RW, bit_US



By **Gray00**

cheatography.com/gray00/

Published 7th June, 2022.

Last updated 27th June, 2022.

Page 2 of 6.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

I/O	Metodi di scorrimento tab_iter	
access (vaddr b, natq dim, bool w, bool s);	Controlla il cavallo di Troia per w(ritable) e s(hared)	E
z = inputx(reg);	Inserisce il valore al registro reg in z	u
outputx(value, reg);	Inserisce nel registro reg il valore value	a
des_x y = new(aligned_val_t{dim}) des_x;	Allinea il nuovo valore in memoria	a
natl spostabili = DIM_PAGINA - (paddr & 0xFFF);	Calcola il numero di byte spostabili per esaurire il frame	V
x: b(byte), l(long), q(uad)		s

Approccio I/O

inspronti e inserimento_lista(pronti, esecuzione)

La primitiva

```
inspro nti();
```

inserisce il processo in esecuzione **in testa** alla coda pronti.

La primitiva

```
inseri men to_ lis ta( pronti, esecuz ione);
```

inserisce il processo in esecuzione nella coda pronti **in maniera ordinata**. Quindi, se nella coda pronti c'è un processo P1 con **priorità uguale** a quella del processo in esecuzione, quest'ultimo verrà inserito nella lista **dopo** il processo P1.

```
inspro nti() ; : esecuzione -> P1
```

```
inseri men to_ lis ta( pronti, esecuz ione); : P1 -> esecuzione
```

Primitive con tipo di ritorno

- Quando viene chiamata l'interruzione?
- Chi prepara il descrittore?
- Cosa deve fare l'interruzione?
- Come rispondiamo all'interruzione?
- Come ci prepariamo alla prossima interruzione?

Nota: nei casi, perlomeno di lettura, potrebbe essere sensato impostare i valori passati come parametri dalla primitiva all'interno del descrittore. Sarà poi l'interruzione (se chiamata ogni tot) a sistemare questi valori già inizializzati con i valori aggiornati. Se invece dovesse essere richiamata solo alla fine del processo, sarà compito della read fare tutto il processo e abilitare le interruzioni alla fine facendo riattivare la `estern_ce`

Nota2: quando in una read viene passato un buffer, questo dovrà essere copiato nel descrittore in quanto è il buffer in cui la periferica dovrà scrivere i valori che dovremo leggere.

Nota3: se il passaggio di byte tra utente e periferica viene effettuato tramite un buffer di appoggio intermedio, dobbiamo necessariamente usare la `memcpy`, in quanto questi dati devono essere copiati in una zona di memoria.

Se abbiamo una primitiva del tipo

```
bool primitiva (parametri)
```

dovremo implementarla in due modi diversi nel modulo sistema e nel modulo I/O.

SISTEMA: non possiamo farle restituire un tipo (deve essere void), quindi il valore di ritorno dovremo inserirlo nel contesto del processo in esecuzione.

```
extern "C" void primitiva (parametri) {
    //operazioni
    esecuzione->contesto[RAX] = valore_ritorno;
}
```

I/O: possiamo utilizzare i tipi di ritorno delle primitive nel seguente modo:

```
extern "C" bool primitiva (parametri) {
    //operazioni
    return valore_ritorno;
}
```

Estern CE

```
// chiamata dalla periferica con un'interruzione
extern "C" void estern_ce (natl_id){

    // recuperare periferica
    for (;;) {
        // sem_wait( mutex);
        // libera il sync occupato nella
        primitiva

        // sem_signal(sync);
        // sem_signal(mutex);
        wfi();
    }
}
```



By Gray00

cheatography.com/gray00/

Published 7th June, 2022.

Last updated 27th June, 2022.

Page 3 of 6.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>