



Departament d'Arquitectura
de Computadors

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Suport en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

Temps real, altres
sistemes

VxWorks

Xenomai

Pthreads RT

Conceptes Avançats de Sistemes Operatius

FIB, DAC, UPC

Curs 2019/20 Q2

Suport per Temps Real

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona



Definicions i conceptes

Definicions i
conceptes

Polítiques de planificació

Polítiques de
planificació

Suport en Linux

Suport en Linux

Inversió de prioritat

Inversió de prioritat

Sistemes per temps real

Sistemes per
temps real

RT-Preempt

RT-Preempt

Temps real, altres sistemes

Temps real, altres
sistemes

VxWorks

VxWorks

Xenomai

Xenomai

Pthreads RT

Pthreads RT

- ▶ Real-time software
 - ▶ must deliver computation results
 - ▶ under application specific time constraints
 - ▶ fails when a result is made available too late
 - ▶ (or too early in some systems)
 - even if the result is otherwise correct

`http://www.on-time.com/rtos-32-docs/rtkernel-32/programming-manual/tasking/real-time.htm`

- ▶ Execució dels processos
 - ▶ A temps, complint el deadline
 - ▶ Sense ser interromputs per altres processos (menys prioritaris)
- ▶ Hard
 - ▶ Perdre un deadline significa una fallada total del sistema
- ▶ Firm
 - ▶ Es poden tolerar algunes pèrdues de deadline
 - ▶ La utilitat dels resultats es nul·la si arriben després del deadline
- ▶ Soft
 - ▶ La utilitat dels resultats decau quan més tard arriben
 - ▶ Linux

http://en.wikipedia.org/wiki/Real-time_computing

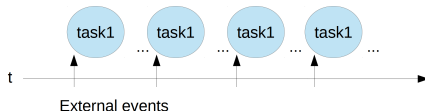
- ▶ Hard
 - ▶ Control del motor d'un cotxe, frens anti-lliscament
 - ▶ Sistema de control d'un avió
 - ▶ Control d'un marcapassos
- ▶ Firm real-time?
 - ▶ Video rendering (també soft?)
- ▶ Soft real-time
 - ▶ Manteniment dels plans de vol de companyies aèries
 - ▶ Latència de segons admissible
 - ▶ Reproducció d'audio i vídeo (també firm?)
 - ▶ Degradació de la qualitat admissible (sala de cinema?)

[http://stackoverflow.com/questions/17308956/
differences-between-hard-real-time-soft-real-time-and-firm-real-time](http://stackoverflow.com/questions/17308956/differences-between-hard-real-time-soft-real-time-and-firm-real-time)

- ▶ Planificació per prioritats
 - ▶ Segons la importància de cada tasca
- ▶ Deadline
 - ▶ Assignat a una tasca, és el temps màxim en el qual la tasca s'ha d'haver executat, per tal que el sistema pugui continuar funcionant
- ▶ Deadline miss
 - ▶ Pèrdua del deadline en una tasca
 - ▶ Les conseqüències poden ser greus hard, firm, soft...

► Tasques periòdiques

- Són aquelles que es repeteixen indefinidament
- Seguint un període d'activació
- Habitualment responen a un event extern

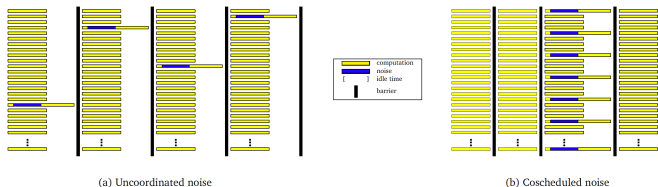


► Tasques aperiòdiques

- Les tradicionals, que comencen i acaben, sense repetir-se necessàriament

- ▶ Jitter: la variació en el temps d'execució d'un procés, deguda a la seva interacció amb
 - ▶ Altres processos
 - ▶ Interrupcions

- ▶ Jitter: la variació en el temps d'execució d'un procés, deguda a la seva interacció amb
 - ▶ Altres processos
 - ▶ Interrupcions



Petrini, F., Kerbyson, D. J., & Pakin, S. (2003, November). The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of ASCI Q. In *SC'03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing* (pp. 55-55). IEEE.
doi: <http://dx.doi.org/10.1145/1048935.1050204>

- ▶ Multitasking
 - ▶ (GP)OS: per donar igual tractament a tots els usuaris/processos/fluxos (fairness, time sharing)
 - ▶ RTOS: usen les prioritats dels processos/fluxos de forma estricta
- ▶ Sobrecàrrega del sistema
 - ▶ OS: permesa
 - ▶ RTOS: no permesa
 - ▶ La sobrecàrrega anirà acompanyada de pèrdues de deadlines

- ▶ Cokernels: al costat del Linux tenim un altre kernel (cokernel, pico-kernel, nano-kernel, o dual kernel)
 - ▶ Capa entre el GPOS i el hardware. Fa servir crides al sistema especialitzades.
 - ▶ *Interrupt dispatcher* i *scheduler*. Molt associat a una versió de kernel de Linux.
 - ▶ Captura les interrupcions i les envia cap a les tasques RT o bé cap al Linux.
 - ▶ Exemples: ADEOS (Xenomai, RTAI) i RTLinux.
- ▶ Patch al kernel de Linux. Un kernel de sol.
 - ▶ Augmenta el grau d'apropiació del codi kernel de Linux actual.
 - ▶ Permet als desenvolupadors explotar l'entorn de programació de Linux
 - ▶ Exemple: PREEMPT_RT

- ▶ Té en compte la prioritat dels processos i el seu estat
 - ▶ *Running* ... en un processador
 - ▶ *Ready* ... no s'executen, però estan a punt
 - ▶ *Suspended* ... (`task_suspend` / `task_resume`)
 - ▶ *Blocked* ... esperant un event
 - ▶ *Timed* ... esperant un event, amb timeout
 - ▶ *Delaying* ... esperant que passi un temps (timer)
- ▶ Els canvis d'estat són produïts per events externs o per una altra task

[http://www.on-time.com/rtos-32-docs/rtkernel-32/
programming-manual/tasking/tasks.htm](http://www.on-time.com/rtos-32-docs/rtkernel-32/programming-manual/tasking/tasks.htm)

Regles que implementa

- ▶ Els N processos (*ready*) amb més prioritat corren
 - ▶ tenint N processadors disponibles
- ▶ Si s'ha de triar entre diversos processos que tenen la mateixa prioritat, s'escull aquell que fa més temps que no s'ha activat
- ▶ Si diversos processos estan esperant (*blocked*) un event, s'activen quan passa l'event respectiu, en l'ordre fixat per la seva prioritat
- ▶ En el moment en que la primera regla no és certa, es fa un canvi de context a un altre procés (més prioritari)

Conceptes Avançats de Sistemes Operatius

- ▶ Per tasques periòdiques
- ▶ La prioritat és estàtica i és la freqüència de la tasca
- ▶ Sempre s'executarà la tasca amb la freqüència més alta

Polítiques de planificació

RT-Preempt

VxWorks

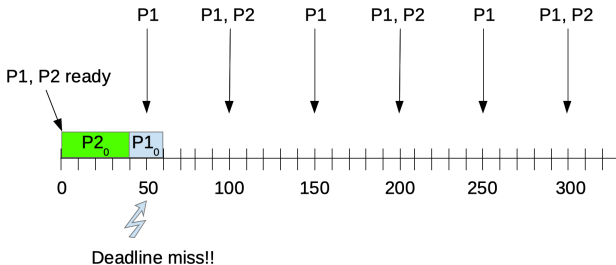
Xenomai

Pthreads RT



Exemple (prioritats manuals)

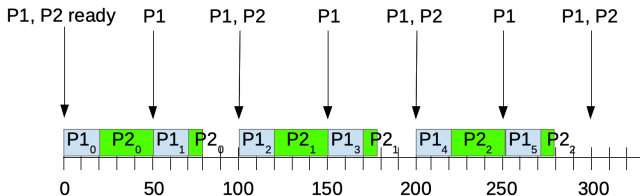
- ▶ P1: freqüència 2/100, durada: 20, prioritat "low"
- ▶ P2: freqüència 1/100, durada: 40, prioritat "high"



- ▶ Lliçó: segons com s'assignen les prioritats, no es compleixen els deadlines

Exemple Rate Monotonic

- ▶ P1: freqüència 2/100, durada: 20, prioritat 2
- ▶ P2: freqüència 1/100, durada: 40, prioritat 1



- ▶ Lliçó: executar primer la que pot tornar a demanar execució abans

- ▶ Earliest Deadline First
 - ▶ Per tasques periòdiques
 - ▶ La prioritat és dinàmica
 - ▶ Depèn de quan proper és el deadline de la tasca
 - Inversament proporcional al temps que falta pel deadline
 - ▶ Cada vegada que la tasca es desbloqueja ha d'indicar el seu deadline al sistema
- ▶ Exemple:
 - P1, prioritat 1/50
 - P2, prioritat 1/100
 - $1/50 > 1/100 \rightarrow$ s'executa primer P1
 - ▶ Representació com en el dibuix anterior

► Soft, linux amb suport per temps real

- Arch Linux

[https:](https://wiki.archlinux.org/index.php/Realtime_process_management)

[//wiki.archlinux.org/index.php/Realtime_process_management](https://wiki.archlinux.org/index.php/Realtime_process_management)

- Debian

http://www.pengutronix.com/software/linux-rt/debian_en.html

http://www.ptxdist.org/software/ptxdist/index_en.html

- Suse <https://www.suse.com/products/realtime/>

- Gentoo

<http://www.gentoo.org/proj/en/desktop/sound/realtime.xml>

- RedHat <http://www.redhat.com/products/mrg/realtime/>

- Ubuntu <https://wiki.ubuntu.com/RealTime>

PAM - pluggable authentication modules

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Support en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

Temps real, altres
sistemes

VxWorks

Xenomai

Pthreads RT

- ▶ Permet configurar els límits dels processos
 - ▶ MEMLOCK, quantitat de memòria virtual que pot fixar-se a memòria física
 - ▶ NICE, màxim valor per a la prioritat d'usuari 0...40
 - ▶ RTPRIO, màxim valor de prioritat de temps real que pot tenir el procés
- ▶ Defineix classes de prioritats per a l'E/S
 - ▶ Realtime: IOPRIO_CLASS_RT, preferent per accedir al disc
 - ▶ Best effort: IOPRIO_CLASS_BE, classe per defecte
- ▶ Arch Linux, Gentoo

- ▶ Inversió de prioritats (*priority inversion*)
 - ▶ Es dóna quan un flux més prioritari espera per entrar en una regió crítica que té un flux menys prioritari
 - ▶ O bé espera per usar un recurs – o dispositiu – que està ocupat per un flux menys prioritari
- ▶ Solució: herència de prioritat
 - ▶ si es dóna el cas, transferir la prioritat del flux més prioritari, al flux menys prioritari per tal que surti de la regió tan aviat com pugui

- ▶ Inversió ilimitada de prioritats (*unbounded priority inversion*)
 - ▶ S'afegeix una tasca (o grup de tasques) intermitja que endarrerix encara més l'execució d'una tasca més prioritària
- ▶ Solucions:
 - ▶ herència de prioritat
 - ▶ sostre de prioritat (*priority ceiling*): cada recurs té una prioritat. Se li assigna un nivell superior (*ceiling*) al de la tasca més prioritària que l'usa. En usar-lo, totes les tasques s'executen en aquest nivell de prioritat

Exemple: Mars Pathfinder

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Suport en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

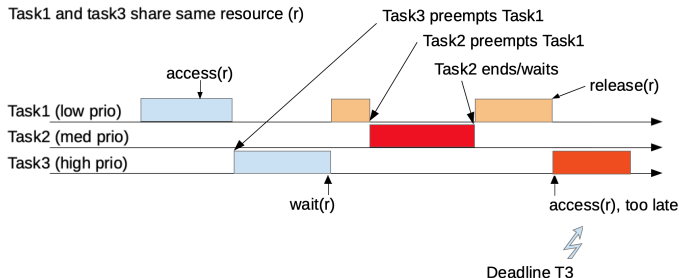
Temps real, altres
sistemes

VxWorks

Xenomai

Pthreads RT

► Inversió ilimitada de prioritats



Font: Operating System Concepts, Silberschatz, Galvin, Gagne, Wiley & Sons, 2010

Exemple: Mars Pathfinder

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Suport en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

Temps real, altres
sistemes

VxWorks

Xenomai

Pthreads RT

- ▶ Tasques en el Mars Pathfinder
 - ▶ Information bus management task high priority
 - ▶ Bloqueja el bus i posa/treu dades
 - ▶ Communications task medium priority
 - ▶ Pot trigar una estona ...
 - ▶ No necessàriament usa el bus
 - ▶ Meteorological data collection task low priority
 - ▶ Bloqueja el bus i hi publica les dades recollides
- ▶ El mutex que protegeix el bus està inicialitzat sense herència de prioritats

<https://sites.fas.harvard.edu/~libe251/fall2019/mars.txt>

Exemple: Mars Pathfinder

- ▶ La solució va consistir en reinicialitzar el mutex amb herència de prioritats

- ▶ VxWorks tenia l'opció de debug activada
- ▶ Això va permetre canviar la inicialització del mutex

The JPL¹ engineers fortuitously decided to launch the spacecraft with this feature still enabled

<https://sites.fas.harvard.edu/~libe251/fall2019/mars.txt>

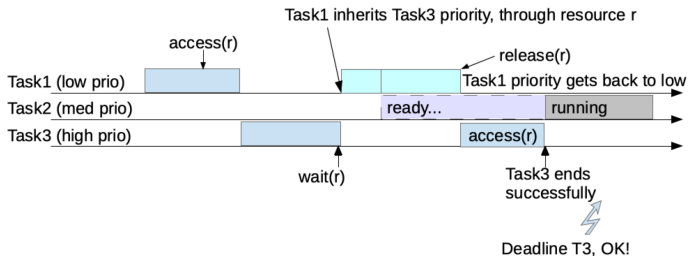
¹Jet Propulsion Laboratory

Exemple: Mars Pathfinder

Font: Operating System Concepts, Silberschatz, Galvin, Gagne, Wiley & Sons, 2010

- Amb herència de prioritats

Task1 and task3 share same resource (r)



Exemple: Mars Pathfinder

► Solució alternativa

- Priority ceiling: assignar prioritats més altes també als recursos

r1	200
r2	150
r3	125
r4	100
MAX_TASK_PRIORITY	99
HIGH_PRIORITY	75
MED_PRIORITY	50
LOW_PRIORITY	25
IDLE_PRIORITY	0

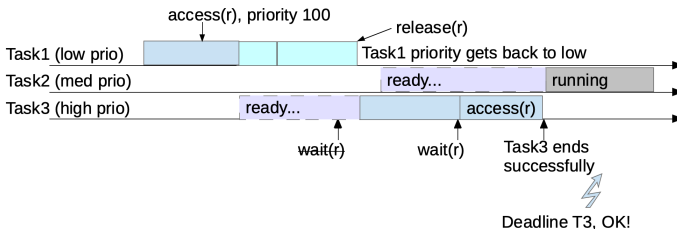
- Les tasques hereden la prioritat del recurs al que accedeixen en exclusió mútua

Font: Operating System Concepts, Silberschatz, Galvin, Gagne, Wiley & Sons, 2010

Exemple: Mars Pathfinder

► Priority ceiling:

Task1 and task3 share same resource (r)



Font: Operating System Concepts, Silberschatz, Galvin, Gagne, Wiley & Sons, 2010

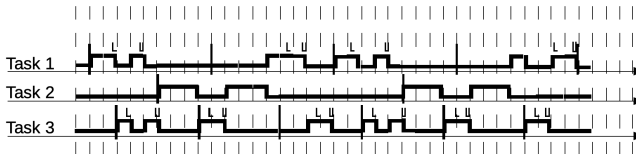
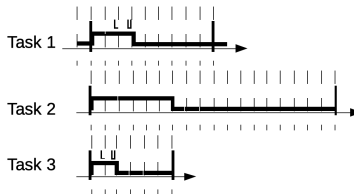
• Control 24-4-2015

Task1 and task3 share same resource (r)

Task1 (low prio)

Task2 (med prio)

Task3 (high prio)



És correcte o hi ha alguna pèrdua de deadline?

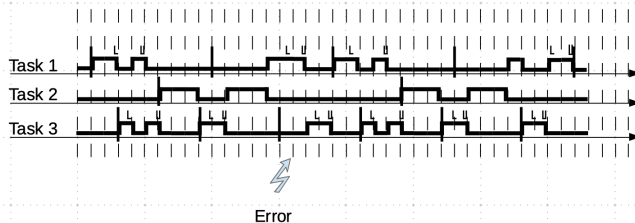
• Control 24-4-2015

Task1 and task3 share same resource (r)

Task1 (low prio)

Task2 (med prio)

Task3 (high prio)



- ▶ Proporciona característiques de hard real time a Linux
 - ▶ Permet que les regions crítiques puguin patir preempcions
 - ▶ Manté les que han de ser no-preemptibles
 - ▶ Implementa herència de prioritats pel kernel
 - ▶ spinlocks
 - ▶ semàfors
 - ▶ Converteix els gestors d'interrupcions en fluxos
 - ▶ Se'ls pot canviar la prioritat
 - ▶ Afegeix rellotges d'alta resolució
- ▶ Debian, Suse, Ubuntu

https://rt.wiki.kernel.org/index.php/PREEMPT-RT_HOWTO

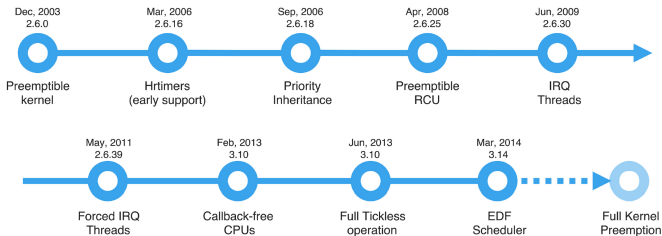
http://taste.tuxfamily.org/wiki/index.php?title=Build_PREEMPT-RT_kernel

- ▶ Incorpora idees de temps real a Linux
 - ▶ Patch al kernel, bastant actualitzat (3.10 → 4.4 Apr 2016)
 - ▶ Suporta herència de prioritats
 - ▶ Soluciona el problema d'inversió de prioritats
- ▶ Xenomai pot basar-se en RT-Preempt
 - ▶ i proporcionar interfícies diferents
 - ▶ VxWorks
 - ▶ QNX
 - ▶ ...
- ▶ <https://www.kernel.org/pub/linux/kernel/projects/rt/>
- ▶ https://rt.wiki.kernel.org/index.php/PREEMPT-RT_HOWTO
- ▶ https://rt.wiki.kernel.org/index.php/CONFIG_PREEMPT_RT_Patch

- ▶ Herència de prioritats en el `pthread_mutex`
 - ▶ Basada en un nou atribut
 - ▶ `PTHREAD_PRIO`
 - `NONE` mutex normal
 - `INHERIT` . . el flux executant la regió crítica ha de tenir la més alta entre les prioritats (seva, dels altres fluxos esperant a qualsevol dels mutex amb `PTHREAD_PRIO_INHERIT` que tingui el flux)
 - `PROTECT` . . .
 - ▶ I la implementació ha d'assegurar que la prioritat es transmet recursivament si el flux que ha hereditat la prioritat es bloqueja en un altre mutex

RT-Preempt

- ▶ Herència de prioritats
 - ▶ PTHREAD_PRIO_PROTECT
 - ▶ usa la prioritat indicada a l'atribut ceiling del mutex
 - ▶ per evitar inversió de prioritats, inicialitzar el ceiling com a més gran o igual a les prioritats dels fluxos que poden agafar aquest mutex
 - ▶ i si em bloquejo en un mutex amb ceiling inferior, mantinc la prioritat superior!!!



Font: Reghenzani, F., Massari, G., & Fornaciari, W. (2019). The real-time linux kernel: A survey on preempt_rt. *ACM Computing Surveys (CSUR)*, 52(1), 1-36. <https://doi.org/10.1145/3297714>

- ▶ Realtime-Preempt Kernel Patch
 - ▶ Deterministic scheduler
 - ▶ Preemption support
 - ▶ PI mutexes
 - ▶ High-Resolution timer
 - ▶ Preemptive Read-Copy update
 - ▶ IRQ threads
 - ▶ Raw Spinlock annotation
 - ▶ Forced IRQ threads
 - ▶ R/W semaphore cleanup
 - ▶ Full Realtime preemption support

<https://www.osadl.org/Realtime-Linux.projects-realtime-linux.0.html>

Deterministic scheduler

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Suport en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

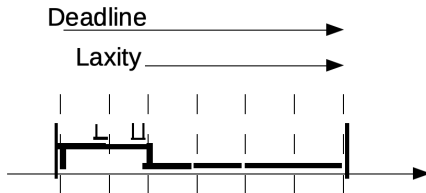
Temps real, altres
sistemes

VxWorks

Xenomai

Pthreads RT

- ▶ Gran número d'alternatives:
 - ▶ EDF, Earliest Deadline First
 - ▶ Schedule the task who's deadline will expire first
 - ▶ Runs work as soon as possible
 - ▶ LLF, Least Laxity First (nom terriblement dolent)
 - ▶ Schedule the task that has least room to still make its deadline
 - ▶ Runs work as late as possible



<https://www.osadl.org/fileadmin/dam/presentations/RTLWS11/peterz-edf-why-not.pdf>
<http://class.ece.iastate.edu/cpre458/cpre558.F00/notes/new-rt-s-lect03.ppt>

► cyclicttest

► Linux 3.10.17-rt12-smp PREEMPT RT – Atom N455

```
$ cyclicttest -a -t -n -p 99
```

```
policy: fifo: loadavg: 1.98 1.60 0.99 2/329 1299
```

```
T: 0 ( 1144) P:99 I:1000 C: 503964 Min: 4 Act: 26 Avg: 33 Max: 3354
```

```
T: 1 ( 1145) P:99 I:1500 C: 335976 Min: 11 Act: 64 Avg: 40 Max: 2535
```

```
policy: fifo: loadavg: 0.65 0.78 0.83 2/355 1339
```

```
T: 0 ( 1301) P:99 I:1000 C: 816333 Min: 10 Act: 22 Avg: 31 Max: 3803
```

```
T: 1 ( 1302) P:99 I:1500 C: 544222 Min: 7 Act: 24 Avg: 33 Max: 2787
```

```
policy: fifo: loadavg: 0.58 0.91 1.16 1/387 3784
```

```
T: 0 ( 1919) P:99 I:1000 C:2363722 Min: 8 Act: 33 Avg: 30 Max: 4550
```

```
T: 1 ( 1920) P:99 I:1500 C:1575814 Min: 7 Act: 35 Avg: 29 Max: 3860
```

► cyclicttest

► Linux 3.9.10-smp SMP – Atom N455

```
$ cyclicttest -a -t -n -p 99
```

```
policy: fifo: loadavg: 0.77 0.58 0.42 2/432 2826
```

```
T: 0 ( 2742) P:99 I:1000 C: 232419 Min: 11 Act: 28 Avg: 32 Max: 10589
```

```
T: 1 ( 2743) P:99 I:1500 C: 154946 Min: 11 Act: 31 Avg: 37 Max: 19370
```

```
policy: fifo: loadavg: 1.68 0.94 0.57 1/377 2871
```

```
T: 0 ( 2828) P:99 I:1000 C: 119921 Min: 10 Act: 22 Avg: 40 Max: 15604
```

```
T: 1 ( 2829) P:99 I:1500 C: 79947 Min: 11 Act: 28 Avg: 32 Max: 11860
```

```
policy: fifo: loadavg: 1.57 1.24 0.77 2/370 2899
```

```
T: 0 ( 2873) P:99 I:1000 C: 234043 Min: 10 Act: 28 Avg: 36 Max: 16849
```

```
T: 1 ( 2874) P:99 I:1500 C: 156029 Min: 10 Act: 21 Avg: 36 Max: 15801
```

► cyclicttest

► Linux 3.10.17-smp SMP – Atom N455

```
$ cyclicttest -a -t -n -p 99
```

```
policy: fifo: loadavg: 0.08 0.17 0.27 2/308 1235
```

```
T: 0 ( 1234) P:99 I:1000 C: 18505 Min: 12 Act: 22 Avg: 49 Max: 16688
```

```
T: 1 ( 1235) P:99 I:1500 C: 12335 Min: 12 Act: 40 Avg: 34 Max: 6511
```

```
policy: fifo: loadavg: 0.47 0.29 0.30 2/309 1239
```

```
T: 0 ( 1238) P:99 I:1000 C: 134952 Min: 11 Act: 26 Avg: 35 Max: 15992
```

```
T: 1 ( 1239) P:99 I:1500 C: 89968 Min: 11 Act: 31 Avg: 30 Max: 4339
```

```
policy: fifo: loadavg: 0.47 0.38 0.33 4/308 1242
```

```
T: 0 ( 1241) P:99 I:1000 C: 258467 Min: 11 Act: 33 Avg: 32 Max: 16334
```

```
T: 1 ( 1242) P:99 I:1500 C: 172311 Min: 10 Act: 67 Avg: 38 Max: 16283
```

► cyclicttest

► Linux 3.2.23 SMP – Core2 Duo P9400

```
$ cyclicttest -a -t -n -p 99
```

```
T: 0 (30397) P:99 I:1000 C: 145316 Min: 4 Act: 18 Avg: 18 Max: 15646  
T: 1 (30398) P:99 I:1500 C: 96877 Min: 4 Act: 13 Avg: 10 Max: 2670
```


► cyclicttest

► Linux 3.2.23 SMP – Core2 Duo P9400

```
$ cyclicttest -a -t -n -p 99
```

```
T: 0 (30397) P:99 I:1000 C: 145316 Min: 4 Act: 18 Avg: 18 Max: 15646  
T: 1 (30398) P:99 I:1500 C: 96877 Min: 4 Act: 13 Avg: 10 Max: 2670
```

► cyclicttest

► Linux 3.14.33 #2 SMP i5-4210U CPU @ 1.70GHz

```
bash-4.3$ sudo ./cyclicttest -a -t -n -p99
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.16 0.08 0.06 2/437 4336
```

T: 0	(4329)	P:99	I:1000	C: 116458	Min: 1	Act: 3	Avg: 2	Max: 184
T: 1	(4330)	P:99	I:1500	C: 77638	Min: 2	Act: 2	Avg: 2	Max: 52
T: 2	(4331)	P:99	I:2000	C: 58229	Min: 1	Act: 5	Avg: 2	Max: 16
T: 3	(4332)	P:99	I:2500	C: 46583	Min: 1	Act: 4	Avg: 2	Max: 54

```
KERNELGIT=git://git.kernel.org/pub/scm/linux/kernel/git
git clone $KERNELGIT/clkwlms/rt-tests.git
```

```
sudo apt install rt-tests
```

► cliclictest

► Linux 4.4.0-Microsoft i7-8565U CPU @ 1.80GHz

```
WARN: stat /dev/cpu_dma_latency failed: No such file or directory
WARN: High resolution timers not available
policy: fifo: loadavg: 0.52 0.58 0.59 2/16 736
T: 0 ( 727) P:99 I:1000 C: 703142 Min: 5 Act: 112 Avg: 310 Max: 18127
T: 0 ( 727) P:99 I:1000 C:4581577 Min: 1 Act: 275 Avg: 321 Max: 85857
T: 1 ( 728) P:99 I:1500 C:3055512 Min: 2 Act: 76 Avg: 325 Max: 84838
T: 2 ( 729) P:99 I:2000 C:2292203 Min: 5 Act: 83 Avg: 323 Max: 85621
T: 3 ( 730) P:99 I:2500 C:1833956 Min: 5 Act: 343 Avg: 325 Max: 85105
T: 4 ( 731) P:99 I:3000 C:1528420 Min: 5 Act: 740 Avg: 322 Max: 84488
T: 5 ( 732) P:99 I:3500 C:1310132 Min: 5 Act: 142 Avg: 326 Max: 83867
T: 6 ( 733) P:99 I:4000 C:1146414 Min: 5 Act: 488 Avg: 329 Max: 85207
T: 7 ( 734) P:99 I:4500 C:1019048 Min: 7 Act: 836 Avg: 325 Max: 84612
```

► cyclicttest, with latency threshold

► Linux 3.14.33 #2 SMP i5-4210U CPU @ 1.70GHz

```
bash-4.3$ sudo ./cyclicttest -a -t -n -p99 -f -b100
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.18 0.13 0.08 2/438 4377
```

```
T: 0 ( 4374) P:99 I:1000 C: 13476 Min: 10 Act: 16 Avg: 20 Max: 77
T: 1 ( 4375) P:99 I:1500 C: 8984 Min: 11 Act: 15 Avg: 18 Max: 80
T: 2 ( 4376) P:99 I:2000 C: 6738 Min: 10 Act: 19 Avg: 18 Max: 80
T: 3 ( 4377) P:99 I:2500 C: 5390 Min: 10 Act: 20 Avg: 19 Max: 63
```

```
# Thread Ids: 04374 04375 04376 04377
```

```
# Break thread: 4374
```

```
# Break value: 372
```

```
bash-4.3$ sudo cat /sys/kernel/debug/tracing/trace
```

```
##### CPU 0 buffer started #####
```

```
<idle>-0 3d... 22890493us!: rcu_eqs_enter_common.isra.48 <-rcu_irq_exit
```

► cyclicttest

- Linux 3.10.58-android-x86+ SMP PREEMPT i5-4210U
CPU @ 1.70GHz

```
[root@localhost rt-tests]# ./cyclicttest -a -t -n -p99  
policy: fifo: loadavg: 0.79 0.82 0.49 2/610 8051
```

```
T: 0 ( 8048) P:99 I:1000 C: 52194 Min: 2 Act: 3 Avg: 2 Max: 20  
T: 1 ( 8049) P:99 I:1500 C: 34796 Min: 2 Act: 3 Avg: 3 Max: 7  
T: 1 ( 8050) P:99 I:2000 C: 26097 Min: 2 Act: 2 Avg: 2 Max: 6  
T: 1 ( 8051) P:99 I:2500 C: 20877 Min: 2 Act: 3 Avg: 3 Max: 7
```

```
[root@localhost rt-tests]# ./signaltest  
0.12 0.51 0.42 1/607 8056
```

```
T: 0 (8055) P: 0 C: 201680 Min: 5 Act: 8 Avg: 9 Max: 280
```

```
[root@localhost rt-tests]# ./pip_stress  
Successfully used priority inheritance to handle an inversion
```

► signaltest

► Linux 3.10.17-rt12-smp SMP PREEMPT RT – Atom

```
$ signaltest
```

```
T: 0 ( 2579) P: 0 C: 851904 Min: 25 Act: 44 Avg: 69 Max: 3537
```

► Linux 3.10.17-smp SMP – Atom

```
$ signaltest
```

```
T: 0 ( 1257) P: 0 C: 744656 Min: 19 Act: 56 Avg: 63 Max: 5329
```

► Linux 3.14.33 #2 SMP i5-4210U CPU @ 1.70GHz

```
$ signaltest
```

```
T: 0 ( 4960) P: 0 C:2140176 Min: 2 Act: 4 Avg: 8 Max: 1969
```

Real-time Ubuntu kernels

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Suport en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

Temps real, altres
sistemes

VxWorks

Xenomai

Pthreads RT

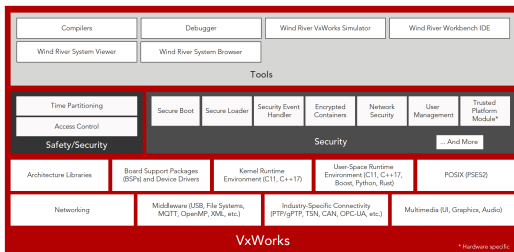
- ▶ lowlatency
 - ▶ soft real-time
 - ▶ basat en el kernel per defecte, amb una configuració agressiva per reduir latències
- ▶ realtime
 - ▶ hard real-time
 - ▶ based on PREEMPT-RT

- ▶ Firm, hard, altres sistemes
 - ▶ Linuxworks LynxOS
 - ▶ WxWorks (Wind River)
 - ▶ QNX Intel, ARM, ... www.qnx.com
 - ▶ HP-RT
 - ▶ Xenomai (Linux)
 - ▶ Microsoft Windows Embedded Compact 2013 x86 i ARM
 - ▶ IntervalZero RTOS platform Windows, www.intervalzero.com
 - ▶ OnTime RTOS-32 platform Win32 apps
 - ▶ RTXC-Quadros Intel, ARM, PPC, ... www.quadros.com

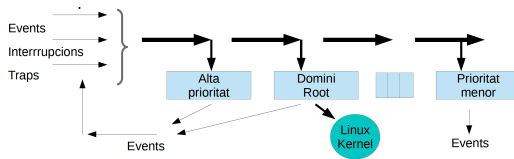
► Recomanacions

- Usar primitives de sincronització de baix nivell
 - operacions atòmiques (`sync_test_and_set`)
 - spin locks (per 10-20 línies de codi)
 - semàfors / mutex
- Usar CPU affinity (`sched_setaffinity`)
- Minimitzar desbalanceig en la feina dels fluxos
 - evitar que hi hagi recursos sobrecarregats

Best Practices: Adoption of Symmetric
Multiprocessing Using VxWorks and
Intel® Multicore Processors

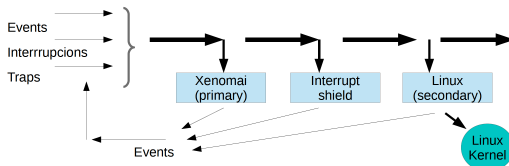


- ▶ Implementa un ordre en la distribució d'events: Adeos
 - ▶ Basat en dominis de protecció
 - ▶ Cadascun amb una prioritat estàtica
 - ▶ Els events es distribueixen primer al domini més prioritari

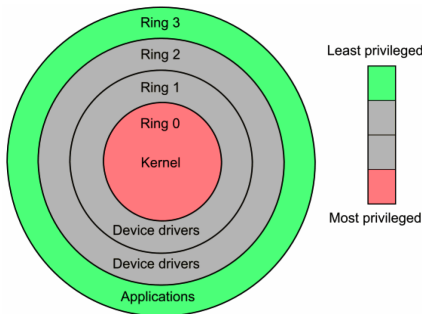


- ▶ Disseny dels dominis
 - ▶ eficients
 - ▶ amb ràfagues curtes d'execució
 - ▶ sobretot els més prioritaris
 - ▶ poden “inhibir” events i/o interrupcions
 - ▶ per exclusió mútua
 - ▶ llavors els events no arriben als dominis menys prioritaris fins que han estat processats

- ▶ Xenomai threads
 - ▶ Poden córrer en mode kernel o en usuari
 - ▶ També en el domini Linux
 - Amb latències més llargues
 - Però sense inversió de prioritats
- ▶ Entorn d'execució "standard" en Xenomai



- ▶ Implementació
 - ▶ aconseguir que Linux cedeixi el control a Adeos
 - ▶ per això es fa que el mòdul d'Adeos mogui Linux al ring 1
 - ▶ Les instruccions privilegiades causen una excepció



► Les instruccions són:

`clts`, clear task-switched flag

`cli`, clear interrupt flag

`in`, input from port

`ins`, input string from port

`invld`, invalidate cache

`lgdt`, load GDT register

`lldt`, load LDT register

`ltr`, load task register

`mov to/from DRn`, move to debug register n

`rdmsr`, read model-specific registers

`rdpmc`, read performance-monitoring counter

`hlt`, halt processor

`sti`, set interrupt flag

`out`, output to port

`outs`, output string from port

`invlpg`, invalidate

`lidt`, load IDT register

`lmsw`, load machine status register

`mov to/from CRn`, move to control register n

`wbinvd`, writeback and invalidate cache

`wrmsr`, write model-specific registers

`rdtsc`, read time-stamp counter

► Com pot permetre Adeos que Linux les executi?

- ▶ Per cada instrucció, es pot executar de diverses maneres:
 - ▶ in, out, ins, outs: són molt usades pels drivers, cal executar-les eficientment
 - ▶ Modificar el descriptor de cada task (procés de Linux) per acomodar el vector de bits que permeti executar-les sobre els ports des del ring 1

- ▶ Per la resta d'instruccions
 - ▶ emulació
 - ▶ l'excepció ha de descodificar la instrucció i modificar l'estat del procés perquè sembli que l'ha executat
 - ▶ execució pas a pas
 - ▶ activa el bit de single *stepping* i deixa executar la instrucció
 - ▶ recupera el control després de la instrucció per desactivar el single *stepping*
 - ▶ no es fa un `sti/cli` sinó que Adeos s'apunta que s'ha fet

Xenomai: exemple

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Suport en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

Temps real, altres
sistemes

VxWorks

Xenomai

Pthreads RT

```
1 #include <stdio.h>
2 #include <signal.h>
3 #include <unistd.h>
4 #include <sys/mman.h>
5 #include <native/task.h>
6 #include <native/timer.h>
7 int main(int argc, char* argv[])
8 {
9     signal(SIGTERM, catch_signal);
10    signal(SIGINT, catch_signal);
11    /* Avoids memory swapping for this program */
12    mlockall(MCL_CURRENT|MCL_FUTURE);
13    /*
14     * Arguments: &task,
15     *             name,
16     *             stack size (0=default),
17     *             priority,
18     *             mode (FPU, start suspended, ...)
19     */
20    rt_task_create(&demo_task, "trivial", 0, 99, 0);
21    /*
22     * Arguments: &task,
23     *             task function,
24     *             function argument
25     */
26    rt_task_start(&demo_task, &demo, NULL);
27    pause();
28    rt_task_delete(&demo_task);
29    return 0;
30 }
31
```

```
1 RT_TASK demo_task;
2 /* NOTE: error handling omitted. */
3 void demo(void *arg)
4 {
5     RTIME now, previous;
6     /*
7      * Arguments: &task (NULL=self),
8      *             start time,
9      *             period (here: 1 s)
10     */
11    rt_task_set_periodic(NULL, TM_NOW, 1000000000);
12    previous = rt_timer_read();
13    while (1) {
14        rt_task_wait_period(NULL);
15        now = rt_timer_read();
16        /*
17         * NOTE: printf may have unexpected impact on the timing of
18         *       your program. It is used here in the critical loop
19         *       only for demonstration purposes.
20         */
21        printf("Time since last turn: %ld.%06ld ms\n",
22              (long)(now - previous) / 1000000,
23              (long)(now - previous) % 1000000);
24        previous = now;
25    }
26 }
```

http://www.xenomai.org/index.php/Xenomai_quick_build_guide

<https://www.rta.org/> – RealTime Application Interface - 4.0 (2013)

Interfície de Xenomai

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Suport en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

Temps real, altres
sistemes

VxWorks

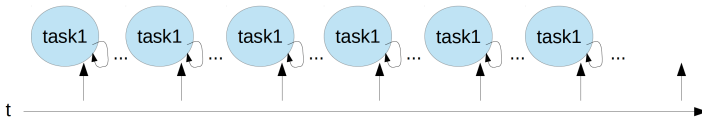
Xenomai

Pthreads RT

```
int rt_task_create (RT_TASK *task, const char *name, int stksize, int prio, int mode)
    Create a new real-time task.
int rt_task_start (RT_TASK *task, void(*entry)(void *cookie), void *cookie)
    Start a real-time task.
int rt_task_suspend (RT_TASK *task)
    Suspend a real-time task.
int rt_task_resume (RT_TASK *task)
    Resume a real-time task.
int rt_task_delete (RT_TASK *task)
    Delete a real-time task.
int rt_task_yield (void)
    Manual round-robin.
int rt_task_set_periodic (RT_TASK *task, RTIME idate, RTIME period)
    Make a real-time task periodic.
int rt_task_wait_period (unsigned long *overruns_r)
    Wait for the next periodic release point.
...
```

- ▶ `rt_task_create()`
 - ▶ Crea i deixa la tasca en suspens
 - ▶ Retorna un handle (`RT_TASK`)
- ▶ `rt_task_start()`
 - ▶ Engega la tasca
- ▶ `rt_task_spawn()`
 - ▶ Combina creació i arrancada

- ▶ `rt_task_set_periodic()`
 - ▶ programa el primer dispar de la tasca i el seu període
 - ▶ Un cop s'ha executat, la tasca s'aturarà en un `rt_task_wait_period()`



Interfície de Xenomai: resum

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Suport en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

Temps real, altres
sistemes

VxWorks

Xenomai

Pthreads RT

- ▶ Gestió de tasques
- ▶ Gestió del temps
- ▶ Sincronització
- ▶ Memòria compartida
- ▶ Pas de missatges
- ▶ Notificacions asíncrones
 - ▶ alarmes i interrupcions
- ▶ Gestors de dispositius
- ▶ Suport a depuració

- ▶ Barriers

- ▶ pthread_barrier_init (&barrier, attr, expected_count)
- ▶ pthread_barrier_wait (&barrier)

```
top - 12:36:23 up 3:50, 9 users, load average: 0.55, 0.41, 0.42
Tasks: 328 total, 3 running, 325 sleeping, 0 stopped, 0 zombie
Cpu0 : 2.2%us, 15.7%sy, 0.0%ni, 82.1%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu1 : 1.8%us, 15.2%sy, 0.0%ni, 82.5%id, 0.0%wa, 0.4%hi, 0.0%si, 0.0%st
Mem: 1935004k total, 1769548k used, 165456k free, 209924k buffers
Swap: 5119996k total, 4696k used, 5115300k free, 824528k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	WCHAN	COMMAND
3792	xavim	20	0	22832	380	304	R	42	0.0	0:07.64	-	barrier
3793	xavim	20	0	22832	380	304	R	41	0.0	0:07.62	futex_wai	barrier
2112	root	20	0	160m	39m	15m	S	1	2.1	2:52.81	poll_sche	X
3777	xavim	20	0	19764	1592	1060	R	1	0.1	0:00.79	-	top
3	root	20	0	0	0	0	S	0	0.0	0:00.26	run_ksoft	ksoftirqd
9	root	20	0	0	0	0	S	0	0.0	0:00.26	run_ksoft	ksoftirqd
2440	xavim	20	0	351m	21m	16m	S	0	1.1	0:00.29	poll_sche	korgac

- ▶ Barriers
 - ▶ Atributs
 - ▶ PTHREAD_PROCESS_PRIVATE
 - ▶ PTHREAD_PROCESS_SHARED


```
union sparc_thread_barrier {
    struct pthread_barrier b;
    struct sparc_thread_barrier_s {
        unsigned int curr_event; // numero de barriers que hem passat
        int lock; // protegeix left
        unsigned int left; // quants en falten per arribar
        unsigned int init_count; // quants n'han d'arribar
        unsigned char pshared; // compartit entre processos o privat
    } s;
};
```

► Inicialització:

```
init_count = nthreads;
left = nthreads
```

Implementació del barrier (I)

```
/* Wait on barrier. */
int
pthread_barrier_wait (barrier)
    pthread_barrier_t *barrier;
{
    union sparc_pthread_barrier *ibARRIER
        = (union sparc_pthread_barrier *) barrier;
    int result = 0;
    int private = ibARRIER->s.pshared ? LLL_SHARED : LLL_PRIVATE;

    /* Make sure we are alone. */
    lll_lock (ibARRIER->b.lock, private);

    /* One more arrival. */
    --ibARRIER->b.left;
```

N threads

1 thread cada cop
left està ben protegit

Implementació del barrier (II)

```

/* Are these all? */
if (ibARRIER->b.left == 0)
{
    /* Yes. Increment the event counter to avoid invalid wake-ups and
       tell the current waiters that it is their turn. */
    ++ibARRIER->b.curr_event;

    /* Wake up everybody. */
    lll_futex_wake (&ibARRIER->b.curr_event, INT_MAX, private);

    /* This is the thread which finished the serialization. */
    result = PTHREAD_BARRIER_SERIAL_THREAD;
}

```

L'últim thread en arribar entra aquí...

... i desperta la resta

... aquest tornarà
PTHREAD_BARRIER_SERIAL_THREAD

Implementació del barrier (III)

Conceptes
Avançats de
Sistemes Operatius

FIB, DAC, UPC

Definicions i
conceptes

Polítiques de
planificació

Suport en Linux

Inversió de prioritat

Sistemes per
temps real

RT-Preempt

Temps real, altres
sistemes

VxWorks

Xenomai

Pthreads RT

```
else
{
    /* The number of the event we are waiting for. The barrier's event
       number must be bumped before we continue. */
    unsigned int event = ibarrier->b.curr_event;

    /* Before suspending, make the barrier available to others. */
    lll_unlock (ibarrier->b.lock, private);

    /* Wait for the event counter of the barrier to change. */
    do
    {
        lll_futex_wait (&ibarrier->b.curr_event, event, private);
        while (event == ibarrier->b.curr_event);
    }
}
```

Tots els altres threads entren aquí

Implementació del barrier (IV)

Conceptes Avançats de Sistemes Operatius

FIB, DAC, UPC

RT-Preempt

VxWorks

Xenomai

Pthreads RT

```

/* Make sure the init_count is stored locally or in a register. */
unsigned int init_count = ibarrier->b.init_count;

/* If this was the last woken thread, unlock. */
if (atomic_increment_val (&ibarrier->b.left) == init_count)
    /* We are done. */
    lll_unlock (ibarrier->b.lock, private);
    L'últim obre el lock per poder tornar-hi
return result;
}

```

L'últim obre el lock per poder tornar-hi

Left està protegit contra decrements pel lock i contra increments per l'operació atòmica

Low level lock

- ▶ Implementat amb lock cmpxchg
 - ▶ Pot comprovar si no hi ha més threads en el procés
 - ▶ i llavors estalvar-se el lock

```
#define lll_trylock(futex) \
({ int ret;
   __asm __volatile (cmpl $0, __libc_multiple_threads(%rip)\n\t"
                     "je 0f\n\t"
                     "lock; cmpxchgl %2, %1\n\t"
                     "jmp 1f\n\t"
                     "0:\tcmpxchgl %2, %1\n\t"
                     "1:"
                     : "=a" (ret), "=m" (futex) : "r" (LLL_LOCK_INITIALIZER_LOCKED), "m" (futex),
                     "0" (LLL_LOCK_INITIALIZER) : "memory");
   ret; })
```

- ▶ `pthread_spin`
 - ▶ `init`
 - ▶ `lock`
 - ▶ `trylock`
 - ▶ `unlock`
- ▶ `pthread_getcpuclockid (pthread_t, *clockid)`

- ▶ Opcions de configuració
 - ▶ CONFIG_HIBERNATION
 - ▶ Permet hibernar en disc (swap area)
 - ▶ CONFIG_RELOCATABLE
 - ▶ Guarda la informació de reubicació, i el kernel es pot carregar a qualsevol posició
 - ▶ CONFIG_RANDOMIZE_BASE
 - ▶ Depends on RELOCATABLE and not (HIBERNATION)
 - ▶ Permet que la càrrega del kernel es faci en una adreça aleatòria
 - ▶ CONFIG_RANDOMIZE_BASE_MAX_OFFSET
 - ▶ 512MB - 1GB