Tutoriel	**AmigaCracking : Robocop**
Protection	**Single Track - Copylock**
Auteur Original	**Scenex**
Soumit par (sur flashtro.com)	**Scenex [2004-07-28]**
Version	**16/03/2018 Gi@nts**
Vérification/Correction	**V2, Testé et fonctionnel de A à Z**

* ROBOCOP CRACK TUTORIEL *

Table des matières

Matériels nécessaire

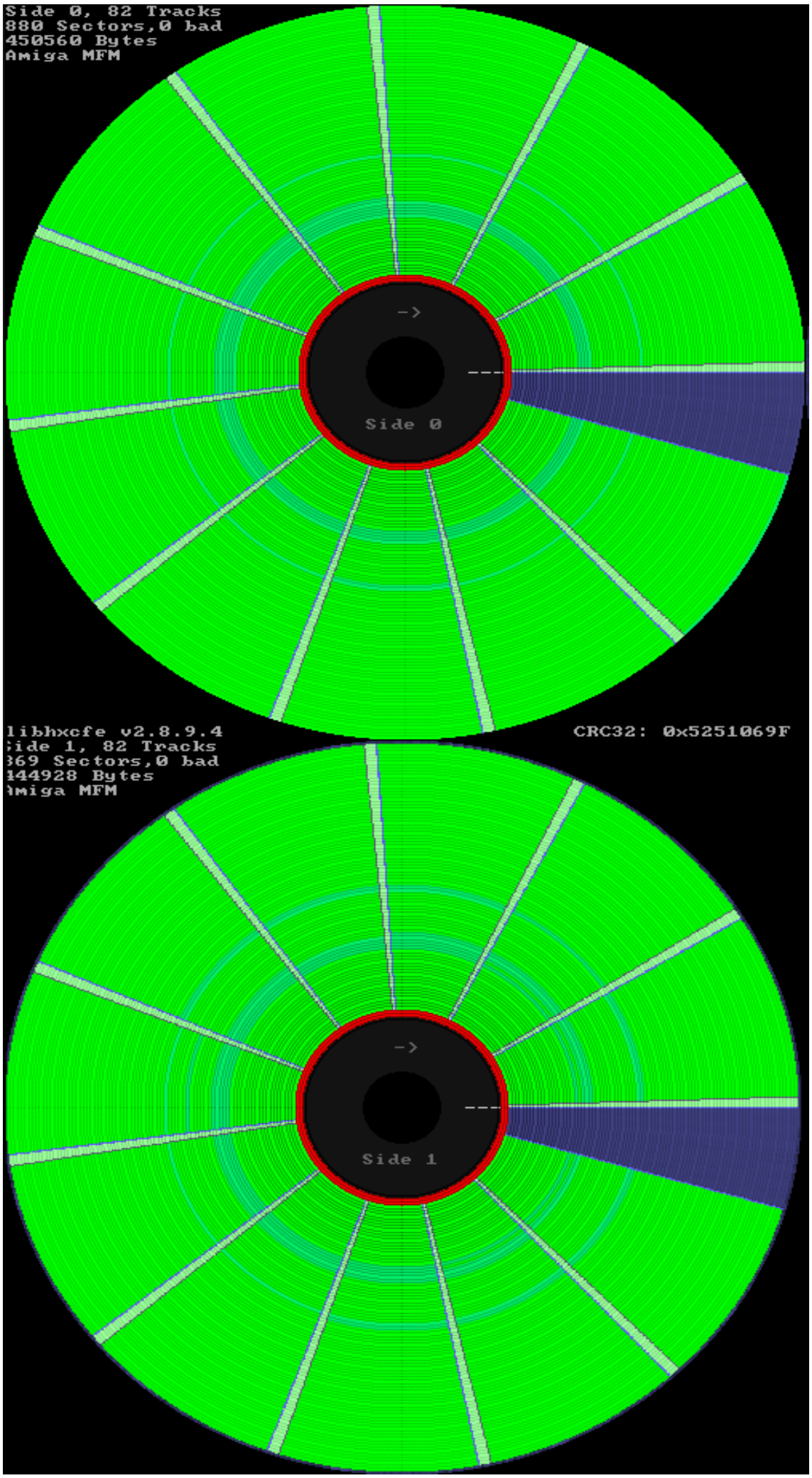
1) Un Amiga avec 512K (ou plus) ou l'émulateur WINUAE
2) Une Carte ACTION REPLAY MKIII (ou ça ROM Image)
3) Le jeu Original Robocop ou son image CAPS (SPS 1620)
4) Le logiciel Xcopy Pro en disquette ou image disk.

Général Info

Ce tutoriel Français est basé sur le tutoriel original de Scenex.
Ce document n'est pas une traduction mot par mot de celui-ci mais plus une nouvelle version.
Suivit pas à pas avec des nouvelles informations.

Bon Tuto.

Gi@nts

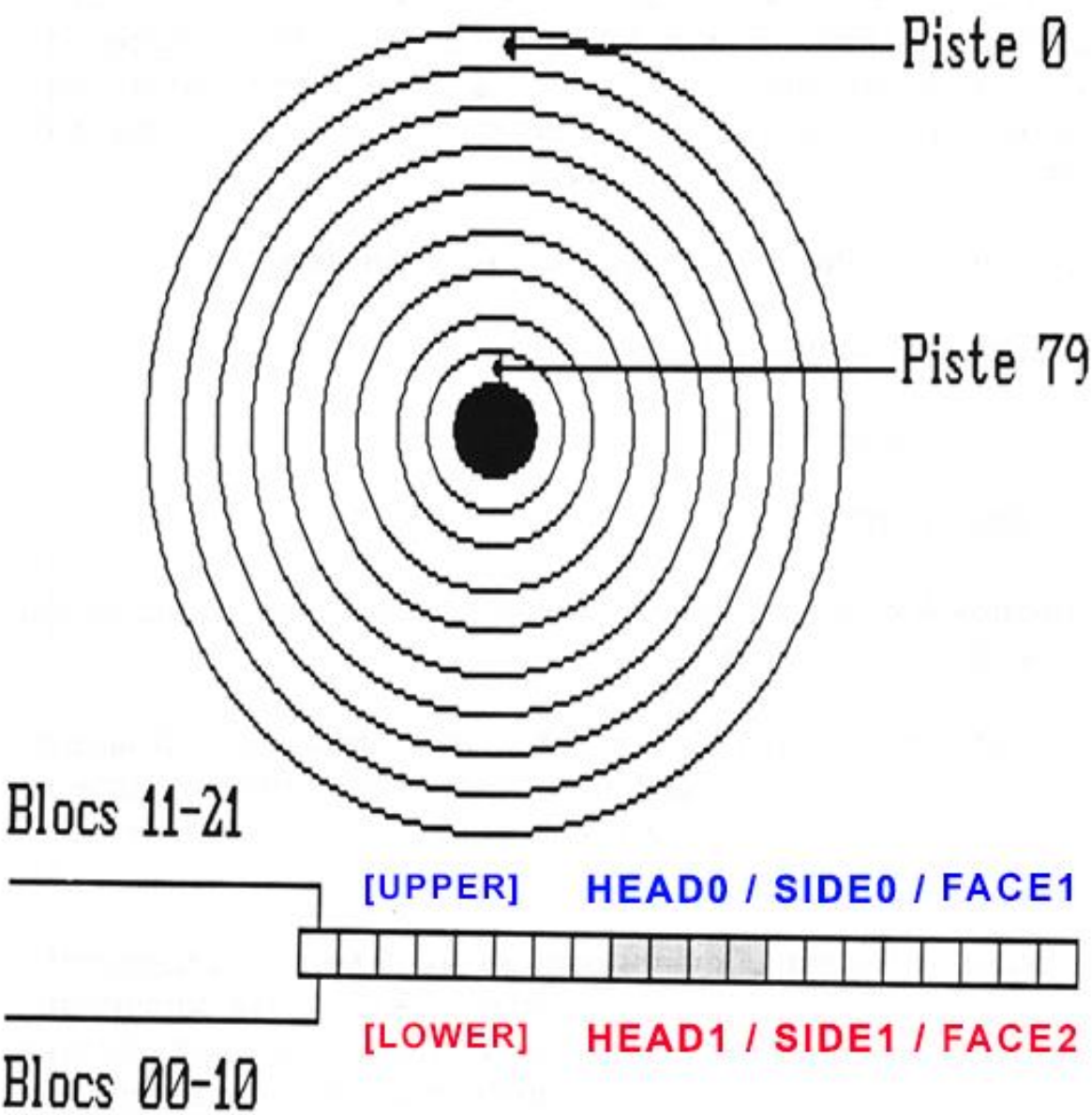

Side 0, 82 Tracks
880 Sectors,0 bad
450560 Bytes
Amiga MFM
->
Side 0
libhxcfe v2.8.9.4
ide 1, 82 Tracks
369 Sectors,0 bad
144928 Bytes
miga MFM
CRC32: 0x5251069F
->
Side 1

Agencement des disquettes Amiga v1.1

En France :

On utilise des termes comme : *piste, bloc, secteur, face*…
Piste : 0 à 79 Certaine disquette pousse jusqu'à 81 voire 82 pistes mais le standard reste quand même 80 pistes (de 0 à 79)
Face : 0/1 ou **1/2** ou **A/B**, Dessus ou dessous tout simplement. Sur Amiga nous avons deux faces utilisées sur 99% des jeux.

Chaque piste, pour un format standard 'AmigaDOS' est composée de plusieurs *bloc ou secteur*, en général 11 **par face**.
Le terme piste peut désigner l'ensemble d'une piste (les deux 'side' du disque), ou uniquement une 'side' d'une piste.
Une piste standard *amigados* est découpée en plusieurs partie appelé **bloc**, **secteur**, **sector**.



Dans d'autre pays :

On utilisera d'autre terme, comme **sector, keys, tracks, cylindre**, **head**…

Le terme **track** par exemple que l'on aurait vite fait de traduire 'piste' ne colle pas forcément à notre description française.
En général, le terme **tracks** désigne toujours une position sur la disquette mais **elle va de 0 à 159** (soit 160 **tracks**)
Le maximum étant 160 et non 80 car on a deux faces bien sûres, en fait, elle correspond à une piste sur une face.

Il peut néanmoins arriver que l'on utilise dans des tuto anglo-saxon le terme *tracks* dans le sens 'piste' en français (donc de 0 à 79 et non de 0 à 159). Mais en règle générale, il a plutôt une plage de 0 à 160.

C'est le terme **cylindre** qui 'colle' plus à notre définition française de **piste.**
En effet, il est courant d'utiliser le terme **cylindre** pour désigner une position sur la disquette de 0 à 79.

Le terme **sector** ou **key** quant à lui corresponds au terme français **bloc** ou **secteur.**

Sur une disquette au format ***Amigados***, nous avons 880ko et nous avons 11 secteurs par face, par piste.

La taille d'une piste ayant une valeur physiquement maximum.
Le nombre maximum de **sector** sur une piste dépends assez logiquement de la taille de ses **sectors.**

Pref…beaucoup de terme qui ne sont pas forcément utilisé dans leur sens propre, le mieux est de lire un tuto et de comprendre quel sens l'auteur a voulu leurs donner.

Il existe aussi un autre type d'appellation utilisé par exemple par le logiciel **MFM-Warp** de Ferox*
**C'est un programme qui scan le disque en bas niveau et essaye d'en réaliser une copie.*

Track	Calcul	Résultat	Format utilisé sous MFMWarp
0	0/2	0 et pair	0.0
1	1/2	0 et impair	0.1
2	2/2	1 et pair	1.0
3	3/2	1 et impair	1.1
156	156/2	78 et pair	78.0
157	157/2	78 et impair	78.1
158	158/2	79 et pair	79.0
159	159/2	79 et impair	79.1

On notera que :

Le premier secteur (secteur 0) appelé aussi *bootbloc* commence sur la *lowerSide* en piste 00
et se fini en piste 79 sur le *upperside*

En *tracks* c'est le même système sauf que l'on terminera en Track 179 et non 79.

La piste Zero est celle situé le plus à l'extérieure du disque.

Le 1er secteur logique, donc le premier bloc sur la disquette, se trouve **piste 0 secteur 0**
Les *bloc* se suivent physiquement mais ne sont pas forcément ordonnée, on parle aussi d'entrelaceme
nt.

Le bloc 11 (si on part de 0 bien sur) n'est pas le 1er secteur de la seconde piste mais le 1er secteur *de la face suivante.*
(voir image ci-dessus)

En format **Amigados**, **la taille d'un secteur est de 512 octets**
Ce qui nous donnes comme taille disponible : 512*11 secteurs*80 pistes*2 faces = 901 120 octets soit 880Ko
Une 'track' AmigaDos a une taille de 512 * 11 = **5632** en décimal soit ***$1600*** *octets*

Mise en application sous l'AR :

Il existe deux commandes sous l'AR qui permettent de charger sauver des pistes, à savoir : **RT** et **WT**
Elles fonctionnent pareil.
L'une permet la lecture, l'autre l'écriture.

#**RT** alias Read Track. Permet le chargement de donnée située sur la disquette vers la mémoire.
#la première valeur sera la track de **départ** [0 à 159] à indiquer en hexa. **/!\ ne pas confondre avec piste**

#La seconde valeur sera le nbr de demi track à copié à partir de là.

#**WT** alias Write Track. Permet la sauvegarde de donnée située en mémoire vers la disquette.

Exemples :

```
RT 20 1 50000
```
Start Track = $20 et taille à lire = 1
On copiera donc la piste !16 (en décimal) side 0 en mémoire $50000

Oui car **20** est donné en hexa, ce qui nous donnes !32 en décimal **mais** il indique une track (de 0 à 159) **PAS en piste.**
Pour avoir l'équivalent en piste on divisera donc par 2 (car deux faces).

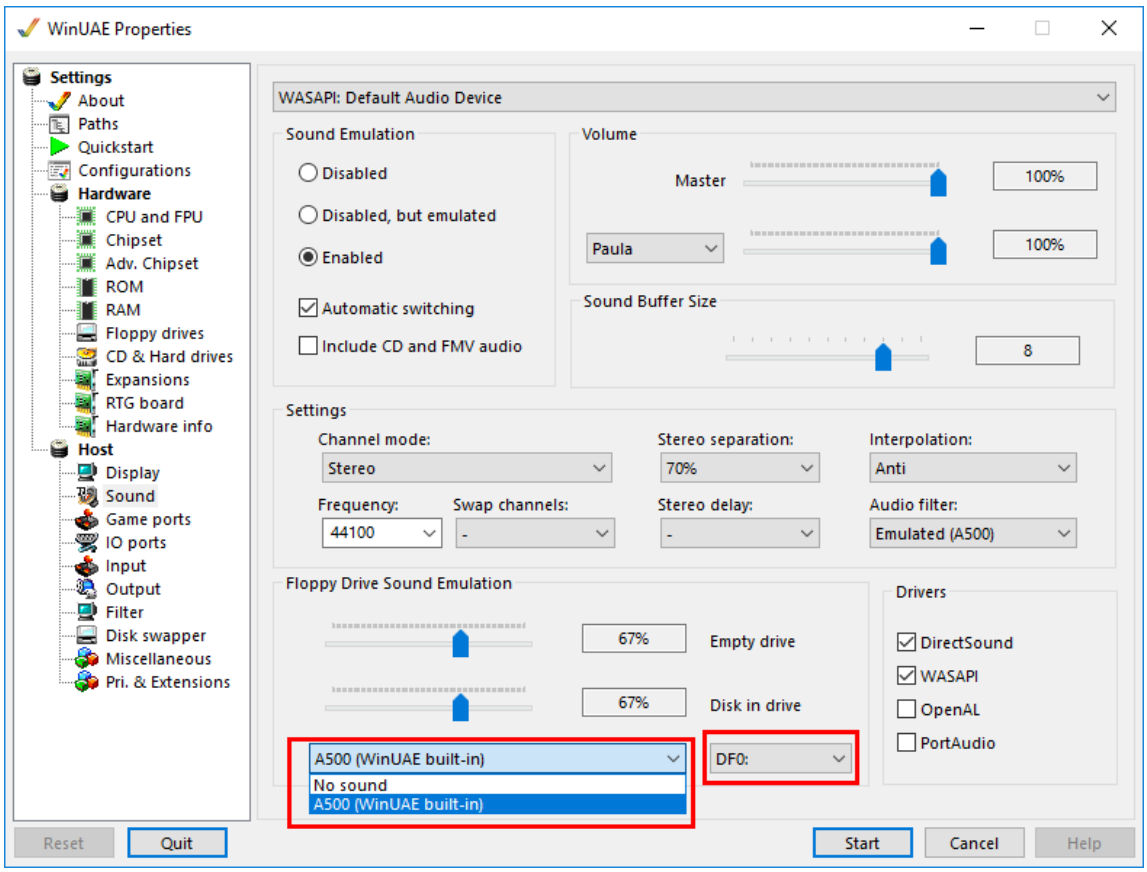
$20/2=$10 = !16 (en décimal donc) et comme il n'y a pas de retenu on est sur la face0.

```
RT 21 2
```
Start Track = $21 et taille à lire = 2
On copiera la piste !16 side 1 et la piste !17 side 0 en mémoire 50000

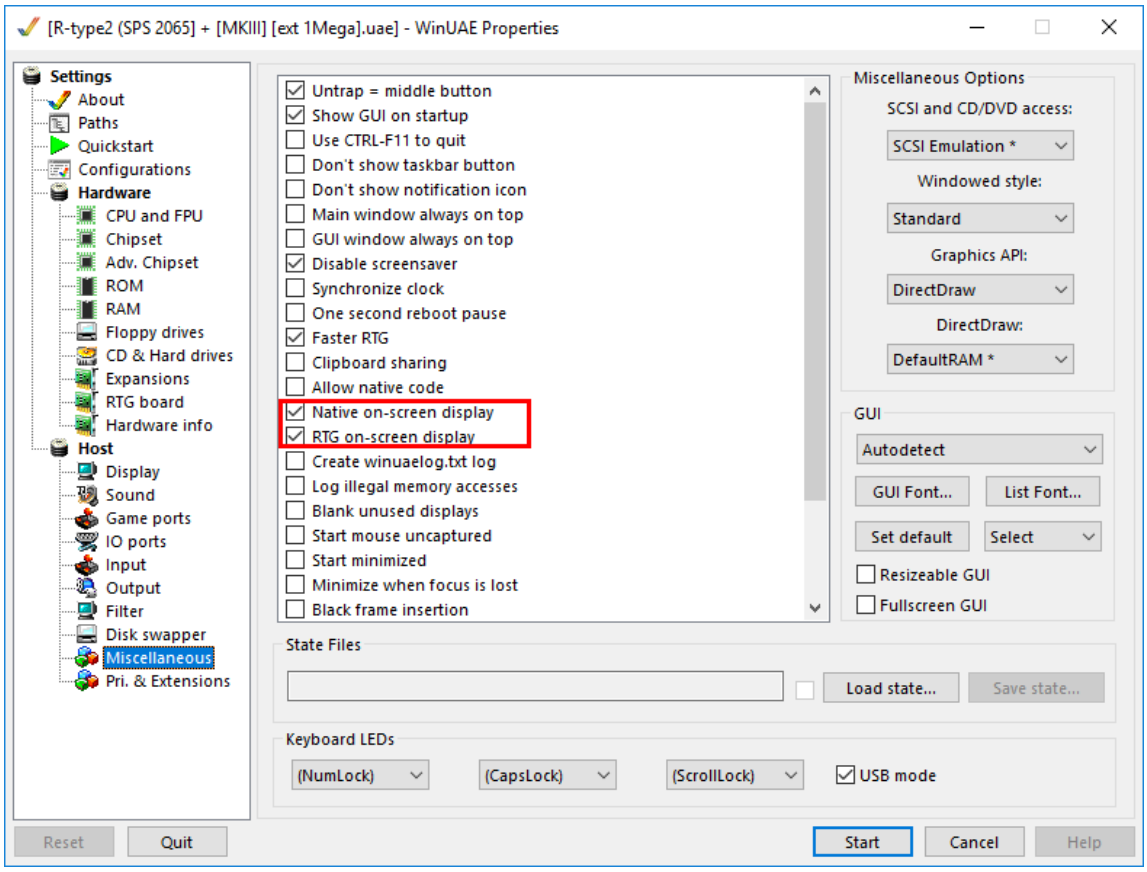
21 est donné en hexa **donc $21 = !33** en décimal.
33/2 = 16.5 ,donc piste 16 side 1 et comme on continue à lire/copier les donnes (**taille à lire =2**), on continue la copie.
On change donc de track car on est déjà sur la face 1 (il existe que 2 faces sur une disquette)
On arrive donc sur la prochaine track à savoir, piste 17 en side 0 puisque que c'est la première face au niveau structure la side 0.

WinUAE

Pour ceux qui utilisent ***winUAE*** pour ces tutoriels (j'imagine, la plupart des personnes), Je vous conseille fortement d'activer le son des lecteurs de disquette histoire d'entendre ce que le lecteur effectue comme accès.
HOST -> SOUND -> FLOPPY DRIVE SOUND EMULATION - > DF0 Built-In

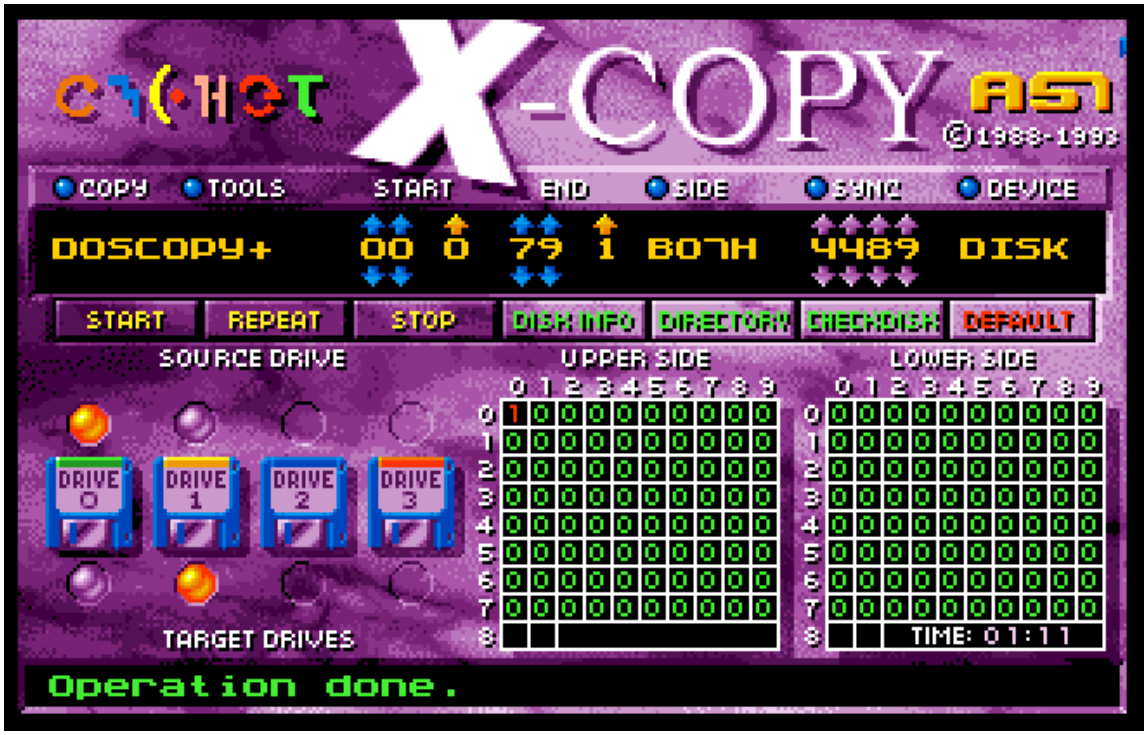


Voir même, pour plus d'information. Par exemple afficher sur qu'elle face l'on se trouve, d'activer :
Host -> Miscellaneous -> Native on-screen display AND RTG on-screen display



Part 1–X-Copy

Comme dans Tous bon hack qui se respecte, on va commencer par essayer de copier le disk original.
Démarrer votre logiciel de copie préféré, à savoir **Xcopy Pro**
Choisissez le mode **DOSCOPY+, insérer** une disquette **vierge** en **DF1** et la disquette du **jeu original** en **DF0**
Lancer la copie



On voie clairement que ça s'annonce mal en piste 0, le jeu ne fonctionnera pas.
Mais gardons quand même sous le coude ce ***backup.***

Rappel des codes d'erreur de Xcopy :

1. *Less or more than 11 sectors*
2. *No sync found*
3. *No sync after gap found*
4. *Header checksum error*
5. *Error in header/format long*
6. *Data block checksum error*
7. *Long track*
8. *Verify error*

Part 2 Analyse de l'image IPF

FILENAME	1620_RoboCop_AMIGA.ipf
TYPE	Floppy_Disk
ENCODER	CAPS(V1)
FILE	1620(V1)
DISK	0
TRACK	00-81
SIDE	0-1
PLATFORM	Amiga
REVOLUTION	4
PROTECTION	COPYLOCK [T00.1]

Mon script d'analyse nous indique la présence d'une protection **Copylock**
Si on regarde la piste 00.1 de plus près, on voie que dans le champ Density est enregistré l'info **Copylock_Amiga**

TrackNumber	Size Record (bytes)	Crc	Status	Track Size	Detail Tr. Size	Start Byte	Bit	DataKey	Block	Density	Signal	Encoder	Flag
T00.0	80	E529516A	Good	12533	100264 bits - Data-95744 + Gap-4520	298	2390	001	11	Auto	cell_2us	0	None
T00.1	80	77E87E38	Good	12535	100264 bits - Data-80992 + Gap-9272	136	1091	002	11	Copylock_Amiga	cell_2us	0	None
T01.0	80	26D0B60A	Good	12532	100256 bits - Data-95744 + Gap-4512	300	2403	003	11	Auto	cell_2us	0	None
T01.1	80	C8D3B462	Good	12533	100264 bits - Data-95744 + Gap-4520	296	2372	004	11	Auto	cell_2us	0	None
T02.0	80	22D1461B	Good	12532	100256 bits - Data-95744 + Gap-4512	300	2406	005	11	Auto	cell_2us	0	None
T02.1	80	4BA66185	Good	12533	100264 bits - Data-95744 + Gap-4520	295	2366	006	11	Auto	cell_2us	0	None
T03.0	80	D75287F7	Good	12532	100256 bits - Data-95744 + Gap-4512	299	2396	007	11	Auto	cell_2us	0	None
T03.1	80	31AA3FD3	Good	12533	100264 bits - Data-95744 + Gap-4520	295	2366	008	11	Auto	cell_2us	0	None
T04.0	80	BF3EA8F9	Good	12533	100264 bits - Data-95744 + Gap-4520	300	2403	009	11	Auto	cell_2us	0	None
T04.1	80	733F74AE	Good	12533	100264 bits - Data-95744 + Gap-4520	295	2362	010	11	Auto	cell_2us	0	None
T05.0	80	11604C7B	Good	12532	100256 bits - Data-95744 + Gap-4512	299	2397	011	11	Auto	cell_2us	0	None
T05.1	80	AD9B2732	Good	12533	100264 bits - Data-95744 + Gap-4520	295	2367	012	11	Auto	cell_2us	0	None
T06.0	80	EBCA62CD	Good	12532	100256 bits - Data-95744 + Gap-4512	299	2397	013	11	Auto	cell_2us	0	None
T06.1	80	0484BE57	Good	12533	100264 bits - Data-95744 + Gap-4520	294	2353	014	11	Auto	cell_2us	0	None
T07.0	80	2E995C03	Good	12532	100256 bits - Data-95744 + Gap-4512	297	2382	015	11	Auto	cell_2us	0	None
T07.1	80	CA8F46DB	Good	12533	100264 bits - Data-95744 + Gap-4520	294	2353	016	11	Auto	cell_2us	0	None
T08.0	80	8D5A7506	Good	12533	100264 bits - Data-95744 + Gap-4520	298	2390	017	11	Auto	cell_2us	0	None
T08.1	80	A19B76B4	Good	12533	100264 bits - Data-95744 + Gap-4520	294	2354	018	11	Auto	cell_2us	0	None
T09.0	80	AE634D83	Good	12532	100256 bits - Data-95744 + Gap-4512	299	2398	019	11	Auto	cell_2us	0	None
T09.1	80	9728B589	Good	12533	100264 bits - Data-95744 + Gap-4520	294	2358	020	11	Auto	cell_2us	0	None
T10.0	80	12F62F5E	Good	12533	100264 bits - Data-95744 + Gap-4520	299	2396	021	11	Auto	cell_2us	0	None
T10.1	80	4E72460B	Good	12533	100264 bits - Data-95744 + Gap-4520	294	2357	022	11	Auto	cell_2us	0	None
T11.0	80	07B53157	Good	12533	100264 bits - Data-95744 + Gap-4520	299	2395	023	11	Auto	cell_2us	0	None
T11.1	80	52873CBC	Good	12533	100264 bits - Data-95744 + Gap-4520	295	2364	024	11	Auto	cell_2us	0	None
T12.0	80	C9268DD7	Good	12533	100264 bits - Data-95744 + Gap-4520	299	2395	025	11	Auto	cell_2us	0	None
T12.1	80	D8C38FEA	Good	12532	100256 bits - Data-95744 + Gap-4512	294	2354	026	11	Auto	cell_2us	0	None
T13.0	80	C187FADB	Good	12533	100264 bits - Data-95744 + Gap-4520	299	2394	027	11	Auto	cell_2us	0	None

TRACK		Data Length (bytes)		Data (bits)				CRC32 of the complete Extra Data Block			Adress
Data block Description	Sector ID	Data		bytes/sector	GAP		Codage	GapDef	DataOff		Adresse
		MFM bits	bytes		MFM bits	bytes			MFM bits	bytes	
[T00.0]		6446		51568				D7A5BFA9			13256-19701
#0	2	8704	545	512	0	1	MFM	0352	0352	15	13256-13287
#1	3	8704	545	512	0	1	MFM	0906	0906	57	13288-13319
#2	4	8704	545	512	0	1	MFM	1460	1460	92	13320-13351
#3	5	8704	545	512	0	1	MFM	2014	2014	126	13352-13383
#4	6	8704	545	512	0	1	MFM	2568	2568	161	13384-13415
#5	7	8704	545	512	0	1	MFM	3122	3122	196	13416-13447
#6	8	8704	545	512	0	1	MFM	3676	3676	230	13448-13479
#7	9	8704	545	512	0	1	MFM	4230	4230	265	13480-13511
#8	10	8704	545	512	0	1	MFM	4784	4784	300	13512-13543
#9	0	8704	545	512	0	1	MFM	5338	5338	334	13544-13575
#10	1	8704	545	512	4520	283	MFM	5892	5892	369	13576-13607
[T00.1]		6138		49104				71230534			19730-25867
#0	N/A	8272	518	N/A	720	46	MFM	0352	0352	15	19730-19761
#1		8272	518		720	46	MFM	0878	0878	55	19762-19793
#2		8272	518		720	46	MFM	1404	1404	88	19794-19825
#3		8272	518		720	46	MFM	1930	1930	121	19826-19857
#4		8272	518		715	45	MFM	2456	2456	154	19858-19889
#5		8272	518		748	47	MFM	2982	2982	187	19890-19921
#6		8272	518		710	45	MFM	3508	3508	220	19922-19953
#7		8272	518		720	46	MFM	4034	4034	253	19954-19985
#8		8272	518		720	46	MFM	4560	4560	286	19986-20017
#9		8272	518		720	46	MFM	5086	5086	318	20018-20049
#10		8272	518		2059	129	MFM	5612	5612	351	20050-20081
[T01.0]		6446		51568				62D5718A			25896-32341
#0	0	8704	545	512	0	1	MFM	0352	0352	15	25896-25927
#1	1	8704	545	512	0	1	MFM	0906	0906	57	25928-25959
#2	2	8704	545	512	0	1	MFM	1460	1460	92	25960-25991

On peut voir que la piste T00.1 sort du format ordinaire AmigaDOS.
Que sa taille de piste est de **6138 bytes** alors que toutes les autres pistes font **6446 bytes.**

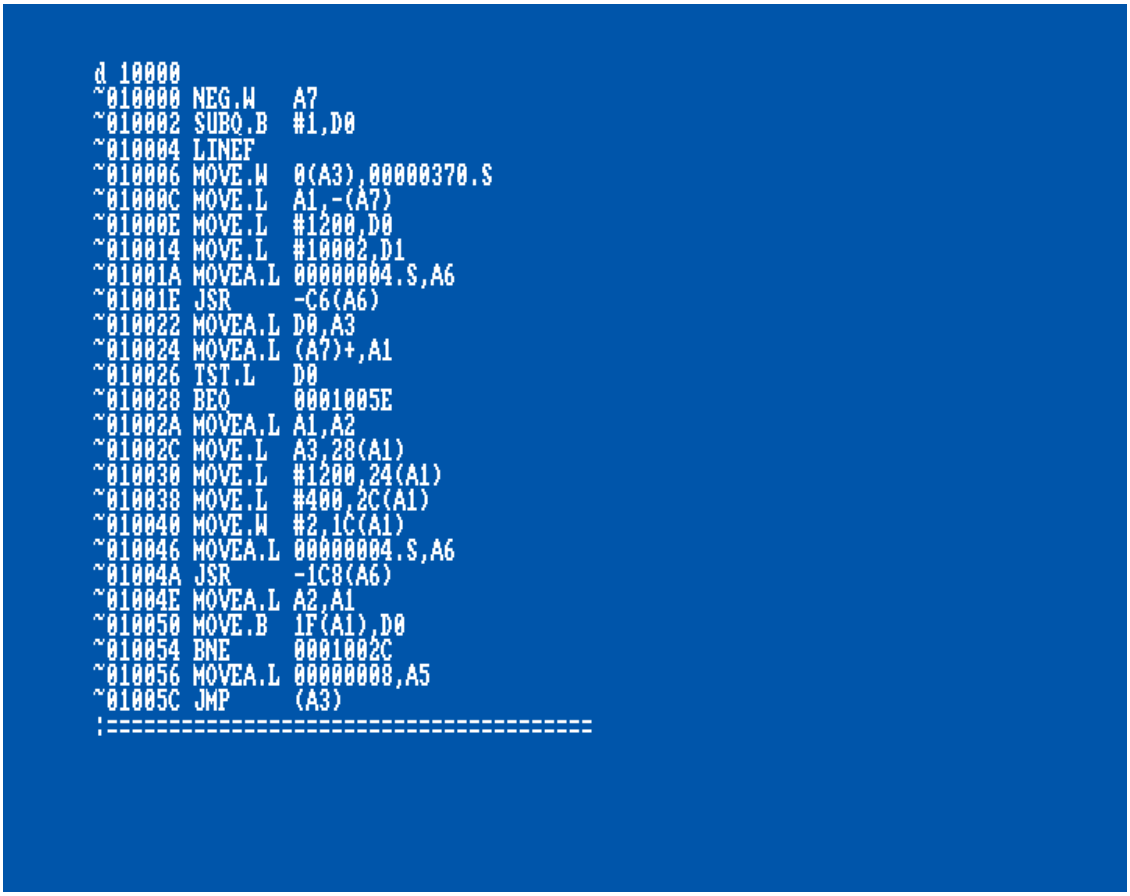
TRACK	Data Length (bytes)	Data (bits)	CRC32 of the complete Extra Data Block	Adress
[T00.0]	6446	51568	D7A5BFA9	13256-19701
[T00.1]	6138	49104	71230534	19730-25867
[T01.0]	6446	51568	62D5718A	25896-32341
[T01.1]	6446	51568	9A98BE8E	32370-38815
[T02.0]	6446	51568	492E8FD1	38844-45289
[T02.1]	6446	51568	B16340D5	45318-51763
[T03.0]	6446	51568	E6474263	51792-58237
[T03.1]	6446	51568	1E0A8D67	58266-64711
[T04.0]	6446	51568	3D273C05	64740-71185
[T04.1]	6446	51568	B053EE19	71214-77659
[T05.0]	6446	51568	E777ECAF	77688-84133
[T05.1]	6446	51568	1F3A23AB	84162-90607
[T06.0]	6446	51568	7DBBCC1A	90636-97081
[T06.1]	6446	51568	1F28A933	97110-103555
[T07.0]	6446	51568	747B4CBD	103584-110029
[T07.1]	6446	51568	ABFEBC1D	110058-116503
[T08.0]	6446	51568	900C52C9	116532-122977
[T08.1]	6446	51568	B54B3982	123006-129451
[T09.0]	6446	51568	26365CA6	129480-135925
[T09.1]	6446	51568	7CDBABE2	135954-142399
[T10.0]	6446	51568	06DDCA6D	142428-148873
[T10.1]	6446	51568	E69AEA77	148902-155347
[T11.0]	6446	51568	9DEE8DF0	155376-161821
[T11.1]	6446	51568	DD6E35DE	161850-168295
[T12.0]	6446	51568	497D7173	168324-174769
[T12.1]	6446	51568	68ABB848	174798-181243

Part 3 Let's do it

Insérer la disquette originale du jeu dans le lecteur de l'Amiga, nous allons charger le ***bootblock*** en mémoire et regarder ça de plus près :

#RT alias Read Track, permet le chargement de la track 0 à 1 (1ère piste de la face 0)
#D, alias Désassemble

Taper : `rt 0 1 10000` puis `d 10000`

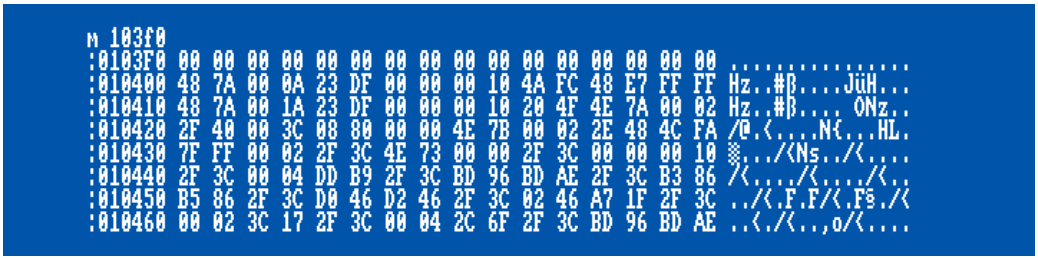


Il semblerait qu'on ait notre petite routine de déplacement de donnée ici :

```
01002C MOVE.L   A3,28(A1)       <== Adresse de destination en A3
010030 MOVE.L #1200,24(A1)      <== Nbr de donnée à copier : $1200
010038 MOVE.L #400.2C(A1)       <== Adresse source de lecture : $400
```

Cela va donc copier **$1200** de donnée à partir de **$400** vers l'adresse contenue dans **A3**

Voyons voir de plus près ce qui se trouve en ce moment à cette adresse**.**
#M alias memory read, permet de voir les données en mémoire.
Taper `m 103F0`



On remarquera notre jolie saut à l'adresse **$1005C** une fois les données déplacées.
On va modifier notre petit '**bootblock**' pour éviter ce **JMP** histoire d'en savoir plus.

Taper :
#A, alias Assemble, Instruction qui va permettre de taper du code assembleur.
#BRA, Instruction du 68000 qui permet de se brancher à l'adresse indiqué, ici une boucle sur nous-même.
#BOOTCHK, permet de calculer le checksum d'un bootblock en mémoire
#WT, alias Write Track, permet d'écrire une zone mémoire sur la disquette à l'adresse indiqué en cylindre.

 # On modifie le code à partir de 1005C
`A 1005C`

#On change le RTS par un BRA sur soit même, la fameuse boucle infini.
`BRA 1005C` [**ENTRER**] puis [**ESCAPE**]

#On calcule le nouveau checksum de ce bootblock
`BOOTCHK 10000`

#On sauve le tout sur disquette
Insérer la disquette de **backup** préalablement crée et taper :
`WT 0 1 10000`

```
d 1005c
~01005C JMP     (A3)
a 1005c
^01005C BRA 1005C
^01005E
bootchk 10000
Old checksum was F2B031EB, now is set to E08531EB
wt 0 1 10000
Disk ok
```

Effectuer un **reboot** de votre Amiga tout en laissant la disquette de backup dans celui-ci.

Après un court chargement (celui du **bootblock**), l'Amiga semble ne plus rien faire.
Il est entré dans notre boucle sans fin.
Entrer dans votre **AR**
Taper : `D`

~0015B4 BRA 000015B4

On est bien dans notre boucle sans fin, impeccable.
On va voir si maintenant on arrive à trouver notre petite signature de **'copylock'** en mémoire.
#Ici on cherche les variables $48 $7A qui correspondent en général au début d'une signature de copylock.
Taper `F 48 7A`

```
d
~0015B4 BRA     000015B4

f 48 7a
Search from: 000000 to: C80000
0059E8  0059F8  _
```

Bingo, trouver !

On jette un petit coup d'œil en mémoire à l'adresse indiquée :
#n alias memory read, permet de voir les données en mémoire en ascii.
Taper `n 59E8`

```
n 59e8
.0059E8 Hz..#ß....JüH...Hz..#ß.... ONz../@.(....N{...HL.▒.../(Ns../(....
.005A28 /(..../(..../(..../(.F.F/(.Fß./(..(./(..,o/(....#....$.Iß.\....$
.005A68 r.q./.m.../4m..T/.Ä)aM.T.@Yi.y.@...z..'..\...6.@.wö=....I...\./.
.005AA8 |.aM...8k9/......N.....QAV.*.v.(@r.W@x|...6ü.M..Ö."üdÖ........äd
.005AE8 f.t...q..u.I..i...9C..(..*&!U...[.LKädl.i.9.Y...u..EW.o..B.),..E
.005B28 s.#...s....e..)..F..q."W....ü%..ß../'1f.Oi.R.$$.sJ..;M{.4.......
```

Au passage, on en profite pour regarder la tête du registre **A3**
#R permet d'afficher tous les registres du 68000
Taper `R`

```
r
D0=00000000 0000001F 00000001 00000000  00000000 00000000 FFFFFFFF 0000004B
A0=00C0152A 00C014E2 00C014E2 000059E8  00001558 00FC0818 00C00276 00C80000
PC = 000015B4 USP = 00C014B2 SR = 0014 T=0 S=0 I=000 X=1 N=0 Z=1 V=0 C=0
```

On notera que l'adresse indiqué dans **A3** est exactement celle trouvé dans notre commande de recherche de **'copylock'**

Ceux d'entre vous qui ont déjà réalisé d'autres tuto copylock auront remarqué que l'adresse **59E8** est récurrente
ps : *voir crack-tuto Arkanoid – revenge Of Doh*

Maintenant on va tous simplement continuer l'exécution du **bootblock**
Cela passe bien sûr par la suppression de notre modification.
#JMP est une instruction assembleur qui permet d'effectuer un saut à une adresse.
Taper `A 15B4` puis `JMP (A3)` [ENTRER] puis [ESCAPE]

```
a 15b4
^0015B4 JMP (A3)
^0015B6
x_
```

Et **avant de retourner à l'exécution** de notre code, on **remplace** la **disquette de backup** par la **disquette originale** de notre **jeu** dans le lecteur Amiga, puis :

#X permet le retour au code Amiga en court.
Taper `X`

Laisser le jeu accéder à 1 ou 2 pistes (**pas plus)** et **entrer** dans votre **AR**
On va aller voir si le code à changer.
Taper `n 59E8`

Sur l'image ci-dessous on peut voir la zone mémoire **AVANT** le **JMP(A3)** et **APRES** le **JMP(A3)**

```
n 59e8
.0059E8 Hz..#ß....JüH...Hz..#ß.... ÖNz../@.(....N(...HL.▒.../(Ns../(....
.005A28 /(..../(..../(..../(.F.F/(.Fs./(..(./(..,o/(....#....$.Is.\....$
.005A68 r.q./.n.../4n..T/.A)aM.T.@Yi.y.@...z..'..\...6.@.wö=....|...\./.
.005AA8 |.aM...8k9/......N.....ÖAV.*.v.(@r.W@x|...6ü.M..Ö."üdÖ........äd
.005AE8 f.t...q..u.I..i...9C..(..*&|U...[.LKädl.i.9.Y...u..EW.o..B.),..E
.005B28 s.#...s....e..)..F..q."W....ü%..ß../'1f.Oi.R.$S.sJ..;M(.4.......

n 59e8
.0059E8 A...!...N@.|..3ü▒..ß..Ö.....A..ß..0(..0ü..Ö...3ü. .ß..A..BC... (
.005A28 ...-. Q..üN.p.2(..4(..6(..A.....C...a...f.N.....H.▒üNV.ü8..D..=D
.005A68 .ü=A..=B..=C..-H.ä-I..äX.@..R@=@..p.6.g...ÖA.|..n..........ü...n
.005AA8 ....g..A=A..HA=A..a..T0...r..@.n..o.2...=A..a..vf!......f.a...a.
.005AE8 .Ff(0....n..g.=@..0..........äBn..0....n..`./.a...a... .g r.2...
.005B28 .n....g..I.ü...n...n../A.(N^J.Lß?.Nux.Bn.üBn..Bn..4...a...f...p.
```

A l'évidence, il y a eu du changement, le '**copylock**' a sûrement décodé les données.
Voyons voir jusqu'où vont ces données.

Taper `m 59E8` et **descendez** jusqu'à atteindre ce qui vous semble la fin des données chargée en mémoire
En l'occurrence : **$6262**

```
:006178 4C DF 00 0C 4E 75 48 E7 20 00 74 55 08 39 00 04 Lß..NuH. .tU.9..
:006188 00 BF E0 01 67 10 70 03 72 FF 61 00 00 20 51 CA ....g.p.r.a.. Q.
:006198 FF EC 70 1E 60 10 30 2E FF DC D0 40 41 FA 00 B4 ..p.`.0..ü.@A...
:0061A8 42 70 00 00 70 00 4C DF 00 04 4E 75 2F 00 61 00 Bp..p.Lß..Nu/.a.
:0061B8 00 32 4A 01 6B 04 08 80 00 00 01 08 80 00 00 13 C0 .2J.k...........
:0061C8 00 BF D1 00 08 C0 00 00 13 C0 00 BF D1 00 20 1F ..............  .
:0061D8 61 00 00 44 4E 75 61 00 00 0A 13 C0 00 BF D1 00 a..DNua.........
:0061E8 4E 75 48 A7 60 00 30 2E FF DC 14 39 00 BF D1 00 NuHs`.0..ü.9....
:0061F8 00 02 00 7F 06 00 00 03 01 82 04 00 00 03 D0 40 ...▒...........@
:006208 32 3B 00 50 08 01 00 00 67 04 08 82 00 02 10 02 2;.P....g.......
:006218 4C 9F 00 06 4E 75 61 00 00 20 08 39 00 00 00 BF L...Nua.. .9.....
:006228 EE 01 66 F6 53 80 66 EE 4E 75 08 39 00 00 00 BF ..föS.f.Nu.9....
:006238 EE 01 66 1C 53 80 67 18 13 FC 00 08 00 BF EE 01 ..f.S.g..ü......
:006248 13 FC 00 CC 00 BF E4 01 13 FC 00 02 00 BF E5 01 .ü....ä..ü......
:006258 4E 75 FF FF FF FF FF FF FF FF 00 00 00 00 00 00 Nu..............
```

Dans un premier temps on va sauver tout ça sur une disquette vierge.
Insérez donc une nouvelle disquette formatée ou formaté la avec la commande format.

Taper :
#SM, alias SaveMemory permet donc de sauver une zone mémoire vers un fichier.
`SM track, 59E8 6262`

Soit exactement : **$6262** – **$59E8** = **$87A** de données

Avant le moindre accès disque, **Entrer** dans votre **AR**
Taper :

#LM alias loadMemory. Permet de charger un fichier vers une zone mémoire spécifique.
```
LM track, 10000
```

Remplace maintenant la **disquette de SAUVEGARDE** par la **disquette originale** dans le lecteur Amiga
On va charger en mémoire les premiers pistes de la disquette originale.
Taper :

```
RT 0 2 30000
```

#TRANS, alias transfert. Permet le transfert d'une zone mémoire vers une autre.
#On transfère nos datas décryptée vers la zone tampon en question (30000+400), juste après le bootblock
```
TRANS 10000 1087A 30400
```

Remplace la **disquette du jeu original** par la **disquette de BACKUP** dans le lecteur Amiga, puis :

Taper :
#Et on sauve le tout sur disquette en lieu et place des pistes de boot et copylock.
#WT alias Write Track, permet l'écriture d'une piste à partir d'une zone mémoire.
```
WT 0 2 30000
```

Maintenant je le jeu devrait être fonctionnel et surtout copiable simplement avec xcopy
Redémarrer votre Amiga et apprécier le jeu !